

Proceedings:



IMAGE UNDERSTANDING WORKSHOP

AD-A279 914



DTIC
ELECTE
MAY 31 1994
S G D



Sponsored by:

Defense Advanced Research Projects Agency
Software and Intelligent Systems Technology Office

April 1993

Image Understanding Workshop

Proceedings of a Workshop
held in
Washington, D.C.

April 18 - 21, 1993

Sponsored by:

Defense Advanced Research Projects Agency
Software and Intelligent Systems Technology Office

Accession For	
NTIS	CRA&I <input checked="" type="checkbox"/>
DTIC	TAB <input type="checkbox"/>
Unannounced <input type="checkbox"/>	
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

This document contains copies of reports prepared for the DARPA Image Understanding Workshop. Included are Principal Investigator reports and technical results from both the basic and strategic computing programs within DARPA/SISTO sponsored projects and certain technical reports from selected scientists from other organizations.

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency of the United States Government.

94 5 27 088

117516
94-16122



Distributed by
Morgan Kaufmann Publishers, Inc.
2929 Campus Drive, Suite 260
San Mateo, CA 94403
ISBN 1-55860-298-4
Printed in the United States of America

TABLE OF CONTENTS

Author Index	ix
Foreword.....	xi
Acknowledgment.....	xv
Section I - Principal Investigator Reports	
"Maryland Progress in Image Understanding," Yiannis Aloimonos, Rama Chellappa, Larry S. Davis, and Azriel Rosenfeld, University of Maryland.....	3
"Image Understanding Research at SRI International," Robert C. Bolles and Martin A. Fischler, SRI International.....	15
"Image Understanding Research at CMU," Katsu Ikeuchi, Takeo Kanade, and Steve Shafer, Carnegie Mellon University.....	27
"Progress in Computer Vision at the University of Massachusetts," Allen R. Hanson, Edward M. Riseman, and Charles A. Weems, University of Massachusetts	39
"Progress in Image Understanding at MIT," W.E.L. Grimson, B.K.P. Horn, T. Poggio, and staff, Massachusetts Institute of Technology.....	49
"Image Understanding Research at Columbia University," Peter K. Allen, Terrance E. Boulton, John R. Kender, and Shree K. Nayar, Columbia University	67
"USC Image Understanding Research, 1992 - 1993," G. Medioni, R. Nevatia, and K. Price, University of Southern California.....	83
"Image Understanding at the University of Rochester," Christopher M. Brown and Randal C. Nelson, University of Rochester	93
"Image Understanding Research at GE," J.L. Mundy, General Electric Corporate Research and Development.....	99
"A Conscious Observer: A Coordinated Effort in Computer Vision," R. Bajcsy, D. Metaxas, M. Mintz, and G. Provan, University of Pennsylvania	105
"IU and UI: An Overview of Research During 1991 - 92," Narendra Ahuja and Thomas Huang, University of Illinois	117
"Vision-Based Navigation," Herbert L. Pick, Jr., University of Minnesota; and William B. Thompson, University of Utah.....	127
"Progress in Image Understanding Research at Brown University," David B. Cooper, Thomas L. Dean, and William A. Wolovich, Brown University	133
"GMU Research on Learning and Vision: Initial Results," J. Bala, R. Michalski, and P. Pachowicz, George Mason University	139
"Image Understanding Research at Johns Hopkins," Lawrence B. Wolff, Johns Hopkins University.....	151
"Image Understanding Research at the Georgia Institute of Technology," Daryl T. Lawton, Georgia Institute of Technology	157
"Image Understanding Research at University of Washington," Robert M. Haralick, University of Washington.....	163
"Model-based Recognition of Objects in Complex Scenes," Thomas O. Binford and Tod S. Levitt, Stanford University.....	169

Section II - Benchmark

"Creating Benchmarking Problems in Machine Vision: Scientific Challenge Problems," Oscar Firschein, DARPA; Martin A. Fischler, SRI International; and Takeo Kanade, Carnegie Mellon University	177
--	-----

Section III - RADIUS

"RADIUS: Automating Image Analysis Through Model-Supported Exploitation," Shirley J. Gee and Arthur M. Newman, Hughes Aircraft Company	185
"Model Matching and Extension for Automated 3D Site Modeling," Yong-Qing Cheng, Robert T. Collins, Allen R. Hanson, and Edward M. Riseman, University of Massachusetts	197
"Site-Model-Based Change Detection and Image Registration," R. Chellappa, L.S. Davis, D. DeMenthon, A. Rosenfeld, and Q. Zheng, University of Maryland	205
"Employing Contextual Information in Computer Vision," Thomas M. Strat, SRI International	217
"Research in Automated Analysis of Remotely Sensed Imagery," Steven D. Cochran, Stephen J. Ford, Stephen J. Gifford, Wilson A. Harvey, J. Chris McGlone, David M. McKeown, Michael F. Polis, and Jefferey A. Shufelt, Carnegie Mellon University	231
"Detection of Buildings from Monocular Views of Aerial Scenes Using Perceptual Grouping and Shadows," A. Huertas, C. Lin, and R. Nevatia, University of Southern California	253

Section IV - Unmanned Ground Vehicle (UGV)

"The JISCT Stereo Evaluation," Robert C. Bolles, H. Harlyn Baker, and Marsha Jo Hannah, SRI International	263
"Reconnaissance Surveillance and Target Acquisition Research for the Unmanned Ground Vehicle Program," Oscar Firschein, LTC Erik G. Mettala, DARPA	275

Section V - Image Understanding Environment

"The Image Understanding Environment: Overview," J.L. Mundy, G.E. Corporate Research and Development; and the IUE Committee	283
"The IUE User Interface," David Dai, MaryAnn Frogge, Warren F. Gardner, Daryl T. Lawton, Heather Pritchett, Andrew T. Rathkopf, and Ian Smith, Georgia Institute of Technology; and the IUE Committee	289
"The Image Understanding Environment: Data Exchange," J.L. Mundy, Keith Price, and R. Welty, G.E. Corporate Research and Development; and the IUE Committee	301
"The Image Understanding Environment: Image Features," Keith Price, University of Southern California; and the IUE Committee	311
"Spacial Objects in the Image Understanding Environment," J.L. Mundy, G.E. Corporate Research and Development; and the IUE Committee	317

Section VI - Workshop Reports

"Computational Sensors: A Report from the DARPA Workshop," Ruzena Bajcsy, University of Pennsylvania; and Takeo Kanade, Carnegie Mellon University	335
"NSF/DARPA Workshop on Machine Learning and Vision - A Summary," Ryszard Michalski and Peter Pachowicz, George Mason University; Yiannis Aloimonos and Azriel Rosenfeld, University of Maryland	351
"DARPA Automatic Sensor Interpretation Workshop," Oscar Firschein, DARPA/SISTO	353

Section VII - Young Investigator Reports

Announcement of Graduate Student Reports	357
--	-----

Section VIII - Cartography/Photogrammetry Camera Calibration

"Camera Calibration Using Line Correspondences," Richard Hartley, G.E. Corporate Research and Development	361
"2-D Images of 3-D Oriented Points," David W. Jacobs, NEC Research Institute.....	367
"Localization and Positioning using Combinations of Model Views," Ronen Basri, The Weizmann Institute of Science; Ehud Rivlin, University of Maryland	377
"Dynamic Camera Self-Calibration from Controlled Motion Sequences," Lisa Dron, Massachusetts Institute of Technology	387
"Fast and Robust 3D Recognition by Alignment," T.D. Alter and W. Eric L. Grimson, Massachusetts Institute of Technology	397
"Alignment Using An Uncalibrated Camera System," Peter K. Allen and Billibon H. Yoshimi, Columbia University	411
"Performance Evaluation of Multispectral Analysis for Surface Material Classification," Stephen J. Ford, John B. Hampshire II, and David M. McKeown, Jr., Carnegie Mellon University	421
"Incorporating Vanishing Point Geometry Into a Building Extraction System," J. Chris McGlone and Jefferey A. Shufelt, Carnegie Mellon University	437
"Measuring the Affine Transform - I: Recovering Scale and Rotation," R. Manmatha and John Oliensis, University of Massachusetts	449
"Matching Perspective Views of Coplanar Structures using Projective Unwarping and Similarity Matching," Ross J. Beveridge, Colorado State University; and Robert T. Collins, University of Massachusetts	459
"Automatic Finding of Main Roads in Aerial Images By Using Geometric - Stochastic Models and Estimation," Meir Barzohar and David B. Cooper, Brown University	465

Section IX - Navigation

"MANIAC - A Next Generation Neurally Based Autonomous Road Follower," Todd M. Jochem, Dean. A. Pomerleau, and Charles E. Thorpe, Carnegie Mellon University	473
"Perceptual Aspects of Navigation," Douglas Gentile, Patricia Melendez, Herbert Pick, Douglas Wagner, Dominic Wegesin, and Albert Yonas, University of Minnesota	481
"Landmark Selection for Accurate Navigation," Karen T. Sutherland, University of Minnesota	485
"Vision-Based Localization," Thomas L. Colvin, Lisa B. Dick, Thomas C. Henderson, William B. Thompson, and Carolyn M. Valiquette, University of Utah.....	491
"A simple, cheap, and robust visual navigation system," Ian Horswill, Massachusetts Institute of Technology	499
"Model Extension and Refinement using Landmarks," Allen R. Hanson, Rakesh Kumar, and Harpeet S. Sawhney, University of Massachusetts	507
"Range-Free Qualitative Navigation," David Dai and Daryl T. Lawton, Georgia Institute of Technology	515
"Interactive Model-Based Vehicle Tracking," Warren F. Gardner, Daryl T. Lawton, and Jun-Hoy Kim, Georgia Institute of Technology	527
"Qualitative Environmental Navigation," John R. Kender and Il-Pyung Park, Columbia University	535

Section X- Visual Learning

"Visual Learning of Object Models from Appearance," Hiroshi Murase and Shree K. Nayar, Columbia University	547
--	-----

"Statistical Properties of Learning Recognition Strategies," Bruce A. Draper, University of Massachusetts	557
Section XI - Active Vision Attention and Selective Perception	
"Detecting Activities," Randal Nelson and Ramprasad Polana, University of Rochester	569
"Studying Control of Selective Perception With T-World and TEA," Christopher M. Brown and Raymond D. Rimey, University of Rochester	575
"Exploratory Active Vision," Yiannis Aloimonos and Jean-Yves Herve, University of Maryland	581
"Planning and Selective Perception for Target Retrieval," Theodore Camus, Thomas Dean, and Jonathan Monsarrat, Brown University	593
"Dynamic Sensor Planning," Steven Abrams and Peter Allen, Columbia University; and Konstantinos Tarabanis, IBM T.J. Watson Research Center	599
Section XII - Motion and Image Sequences	
"A feature-based monocular motion analysis system guided by feedback information," Yong Cheol Kim and Keith Price, University of Southern California	611
"Detection and Segmentation of Feature Trajectories in Multiple, Discontinuous Motion Image Sequences," Narendra Ahuja and Srikanth Thirumalai, University of Illinois	621
"Motion constraint patterns," Cornelia Fermuller, University of Maryland	629
"3-D Recovery of Structural and Kinematic Parameters from Long Sequences of Noisy Images," Rama Chellappa and Ting-Hu Wu, University of Maryland	641
"Recognition and Tracking of 3D Objects by 1D Search," Daniel F. DeMenthon, University of Maryland	653
"Point Correspondence and Motion Detection in Long Image Sequences," Rama Chellappa and Qinfen Zheng, University of Maryland	661
"Optical Flow from 1D Correlation: Application to a simple Time-To-Crash Detector," Nicola Ancona, Tecnopolis CSATA Novus Ortus; and Tomaso Poggio, Massachusetts Institute of Technology	673
"A Paraperspective Factorization Method for Shape and Motion Recovery," Takeo Kanade and Conrad J. Poelman, Carnegie Mellon University	683
"Understanding Noise: The Critical Role of Motion Error in Scene Reconstruction," Allen Hanson, John Oliensis, and J. Inigo Thomas, University of Massachusetts	691
"Translational Decomposition of Flow Fields," Warren F. Gardner and Daryl T. Lawton, Georgia Institute of Technology	697
"Structure and Motion from Region Correspondences and Affine Invariants," David B. Cooper and Chi-Yin Lee, Brown University	707
Section XIII - Object Recognition Using Invariance/Constraints	
"Invariant Object Recognition: A Model Evolution Approach," Peter W. Pachowicz, George Mason University	715
"Quasi-invariant properties and 3-D shape recovery of non-straight, non-constant generalized cylinders," Ramakant Nevatia and Mourad Zerroug, University of Southern California	725
"Invariants of Lines in Space," Richard Hartley, G.E. Corporate Research and Development	737
"Chirality Invariants," Richard Hartley, G.E. Corporate Research and Development	745

"Efficient Recognition of Rotationally Symmetric Surface and Straight Homogeneous Generalized Cylinders," David Forsyth, University of Iowa; Jane Liu and Joe Mundy, G.E. Corporate Research and Development; Charlie Rothwel and Andrew Zisserman, Oxford University	755
"Projective Invariant and Structure from Two Perspective/Orthographic Views: Motion and Recognition," Amnon Shashua, Massachusetts Institute of Technology	767
"An Integrated Approach to Object Recognition," Noah S. Friedland, University of Maryland	777
"Recognition with Local and Semi-local Invariants," Ehud Rivlin and Isaac Weiss, University of Maryland	789
"Constraint Processing Applied to Industrial Inspection and Continuous Product Improvement," Joe Mundy and J. Alison Noble, G.E. Corporate Research and Development	801
"Sensor Modeling Markov Random Fields, and Robust Localization for Recognizing Partially Occluded Objects," Katsushi Ikeuchi and Mark D. Wheeler, Carnegie Mellon University	811
"Quasi-Invariants: Theory and Exploitation," Thomas O. Binford and Tod S. Levitt, Stanford University	819
"A Spherical Representation for the Recognition of Curved Objects," H. Delingette, M. Hebert, and K. Ikeuchi, Carnegie Mellon University	831
"Statistical Object Recognition with the Expectation-Maximization Algorithm in Range-Derived Features," William M. Wells III, Massachusetts Institute of Technology	839
"Scalable Data Parallel Geometric Hashing - Experiments of MasPar MP-1 and on Connection Machine CM-5," Ashfaq Khokhar and Viktor K. Prasanna, University of Southern California	851
"An Integrated Object Recognition System Based on High Degree Implicit Polynomials, Algebraic Invariants, and Bayesian Methods," David B. Cooper, Daniel Keren, and Jayashree Subrahmonia, Brown University	861
"Reflectance Based Recognition," Ruud M. Bolle and Shree K. Nayar, Columbia University	867
"Robotic Three Dimensional Dual-Drive Hybrid Control for Tactile Sensing and Object Recognition," Kelly A. Korzenowski and William A. Wolovich, Brown University	873

Section XIV - Segmentation and Partitioning

"Inferring Global Perceptual Contours from Local Features," Gideon Guy and Gerard Medioni, University of Southern California	881
"A Transform for Detection of Multiscale Image Structure," Narendra Ahuja, University of Illinois	893
"Scene Segmentation and Volumetric Descriptions of SHGCs from a Single Intensity Image," Ramakant Nevatia and Mourad Zerroug, University of Southern California	905
"Saliency Detection and Partitioning Planar Curves," Martin A. Fischler and Helen C. Wolf, SRI International	917
"Detecting Occluding Edges Without Computing Dense Correspondence," Lambert E. Wixson, University of Rochester	933
"Robust Shape Recovery from Occluding Contours Using a Linear Smoother," Richard Szeliski, Digital Equipment Corporation; and Richard Weiss, University of Massachusetts	939

Section XV - Depth Analysis

"Integrating Multiple Range Images Using Triangulation," Yang Chen and Gerard Medioni, University of Southern California	951
"Range Estimation From Focus Using a Non-frontal Imaging Camera," Narendra Ahuja and Arun Krishnan, University of Illinois	959
"Depth from Focusing and Defocusing," Steve Shafer and Yalin Xiong, Carnegie Mellon University	967

"A VLSI Smart Sensor for Fast Range Imaging," Andrew Gruss, Takeo Kanade, and Shigeyuki Tada, Carnegie Mellon University.....	977
"3-D Stereo Using Photometric Ratios," Elli Angelopoulou and Lawrence B. Wolff, Johns Hopkins University	987
"Learning and Feature Selection in Stereo Matching," Michael S. Lew, Thomas S. Huang, and Kam W. Wong, University of Illinois at Urbana-Champaign.....	993
"Implementation and Performance of Fast Parallel Multi-Baseline Stereo Vision," Jon A. Webb, Carnegie Mellon University.....	1005

Section XVI - Reflectance Polarization Color

"Estimating Scene Properties from Color Histograms," Carol L. Novak, Siemens Corporate Research; and Steven A. Shafer, Carnegie Mellon University.....	1013
"Diffuse And Specular Reflection From Dielectric Surfaces," Lawrence B. Wolff, Johns Hopkins University.....	1025
"Polarization Camera Technology," Lawrence B. Wolff, Johns Hopkins University.....	1031
"Generalization of the Lambertian Model," Shree K. Nayar and Michael Oren, Columbia University	1037
"Separation of Reflection Components Using Color and Polarization," Terrance Boulton, Xi-Sheng Fang, and Shree K. Nayar, Columbia University	1049

Section XVII - Low-Level Vision

"Local Step Edge Estimation: A New Algorithm, Statistical Model and Performance Evaluation," Thomas O. Binford and Sheng-Jyh Wang, Stanford University	1063
"Performance Characterization of Edge Operators," Robert M. Haralick and Visvanathan Ramesh, University of Washington	1071

Section XVIII - Shape from X

"Shape from Shadows under Error," John R. Kender and David Yang, Columbia University.....	1083
"Shape and Motion from Linear Features," Warren F. Gardner and Daryl T. Lawton, Georgia Institute of Technology	1091
"Object-Centered Surface Reconstruction: Combining Multi-Image Stereo Shading," P. Fua and Y. Leclerc, SRI International.....	1097
"Provably Convergent Algorithms for Shape from Shading," Paul Dupuis, Brown University; and John Oliensis, University of Massachusetts.....	1121

Section XIX - IU Architectures & Software

"Status and Current Research in the Image Understanding Architecture Program," James Burrill, Alfred Hough and Richard Lerner, Amerinex Artificial Intelligence Inc.; Katja Daumüller, Steven Dropsho, Rabbi Dutta, Martin Herboldt, Glen Weaver and Charles Weems, University of Massachusetts.....	1133
"Integrating the Lisp/CLOS-C/C++ Environments: An Approach to Modular Interface Formats," Jon L. White, Lucid, Inc	1141
"Parallel Dense Depth from Motion on the Image Understanding Architecture," R. Dutta, E.M. Riseman and C.C. Weems, University of Massachusetts	1145
"ISR3: A Token Database for Integration of Visual Modules," Bruce Draper, Allen R. Hanson, and Edward M. Riseman, University of Massachusetts	1155

AUTHOR INDEX

Abrams, Steven	599	Gifford, Stephen J.	231
Ahuja, Narendra	117, 621, 893, 959	Grimson, W. Eric L.	49, 397
Allen, Peter K.	67, 411, 599	Gruss, Andrew	977
Aloimonos, Yiannis	3, 351, 581	Guy, Gideon	881
Alter, T.D.	397	Hampshire II, John B.	421
Ancona, Nicola	673	Hannah, Marsha Jo	263
Angelopoulou, Elli	987	Hanson, Allen R.	39, 197, 507, 691, 1155
Bajcsy, Ruzena	105, 335	Haralick, Robert M.	163, 1071
Baker, H. Harlyn	263	Hartley, Richard	361, 737, 745
Bala, J.	139	Harvey, Wilson A.	231
Barzohar, Meir	465	Henderson, Thomas C.	491
Basri, Ronen	377	Hebert, M.	831
Beveridge, Ross J.	459	Herbordt, Martin	1133
Binford, Thomas O.	169, 819, 1063	Herve, Jean-Yves	581
Bolle, Ruud M.	867	Horn, B.K.P.	49
Bolles, Robert C.	15, 263	Horswill, Ian	499
Boult, Terrance E.	67, 1049	Hough, Alfred	1133
Brown, Christopher M.	93, 575	Huang, Thomas S.	117, 993
Burrill, James	1133	Huertas, A.	253
Camus, Theodore	593	Ikeuchi, Katsu	27, 811, 831
Chellappa, Rama	3, 205, 641, 661	Jacobs, David W.	367
Chen, Yang	951	Jochem, Todd M.	473
Cheng, Yong-Qing	197	Kanade, Takeo	27, 177, 335, 683, 977
Cochran, Steven D.	231	Kender, John R.	67, 535, 1083
Collins, Robert T.	197, 459	Keren, Daniel	86
Colvin, Thomas L.	491	Khokhar, Ashfaq	851
Cooper, David B.	133, 465, 707, 861	Kim, Jun-Hoy	527
Dai, David	289, 515	Kim, Yong Cheol	611
Daum Mueller, Katja	1133	Korzeniowski, Kelly A.	873
Davis, Larry S.	3, 205	Krishnan, Arun	959
DeMenthon, Daniel F.	205, 653	Kumar, Rakesh	507
Dean, Thomas L.	133, 593	Lawton, Daryl T.	157, 289, 515, 527, 697, 1091
Delingette, H.	831	Leclerc, Y.	1097
Dick, Lisa B.	491	Lee, Chi-Yin	707
Draper, Bruce A.	557, 1155	Lerner, Richard	1133
Dron, Lisa	387	Levitt, Tod S.	169, 819
Dropsho, Steven	1133	Lew, Michael S.	993
Dupuis, Paul	1121	Lin, C.	253
Dutta, Rabbi	1133, 1145	Liu, Jane	755
Fang, Xi-Sheng	1049	Manmatha, R.	449
Fermuller, Cornelia	629	McGlone, J. Chris	231, 437
Firschein, Oscar	177, 275, 353	McKeown, Jr., David M.	231, 421
Fischler, Martin A.	15, 177, 917	Medioni, Gerard	83, 881, 951
Ford, Stephen J.	231, 421	Melendez, Patricia	481
Forsyth, David	755	Metaxas, D.	105
Friedland, Noah S.	777	Mettala, Erik G.	275
Frogge, MaryAnn	289	Michalski, Ryszard	139, 351
Fua, P.	1097	Mintz, M.	105
Gardner, Warren F.	289, 527, 697, 1091	Monsarrat, Jonathan	593
Gee, Shirley J.	185	Mundy, Joseph L.	99, 283, 301, 317, 755, 801
Gentile, Douglas	481	Murase, Hiroshi	547

Nayar, Shree K.	67, 547, 867, 1037, 1049	White, Jon L.	1141
Nelson, Randal C.	93, 569	Wixson, Lambert E.	933
Nevatia, Ramakant	83, 253, 725, 905	Wolf, Helen C.	917
Newman, Arthur M.	185	Wolff, Lawrence B.	151, 987, 1025, 1031
Noble, J. Alison	801	Wolovich, William A.	133, 873
Novak, Carol L.	1013	Wong, Kam W.	993
Oliensis, John	449, 691, 1121	Wu, Ting-Hu	641
Oren, Michael	1037	Xiong, Yalin	967
Pachowicz, Peter W.	139, 351, 715	Yang, David	1083
Park, Il-Pyung	535	Yonas, Albert	481
Pick, Jr., Herbert L.	127, 481	Yoshimi, Billibon H.	411
Poelman, Conrad J.	683	Zerroug, Mourad	725, 905
Poggio, Tomaso	49, 673	Zheng, Qinfen	205, 661
Polana, Ramprasad	569	Zisserman, Andrew	755
Polis, Michael F.	231		
Pomerleau, Dean. A.	473		
Prasanna, Viktor K.	851		
Price, Keith	83, 301, 311, 611		
Pritchett, Heather	289		
Provan, G.	105		
Ramesh, Visvanathan	1071		
Rathkopf, Andrew T.	289		
Rimey, Raymond D.	575		
Riseman, Edward M.	39, 197, 1145, 1155		
Rivlin, Ehud	377, 789		
Rosenfeld, Azriel	3, 205, 351		
Rothwell, Charlie	755		
Sajwheey, Harpeet S.	507		
Shafer, Steven A.	27, 967, 1013		
Shashua, Amnon	767		
Shufelt, Jefferey A.	231, 437		
Smith, Ian	289		
Strat, Thomas	217		
Subrahmonia, Jayashree	861		
Sutherland, Karen T.	485		
Szeliski, Richard	939		
Tada, Shigeyuki	977		
Tarabanis, Konstantinos	599		
Thirumalai, Srikanth	621		
Thomas, J. Inigo	691		
Thompson, William B.	127, 491		
Thorpe, Charles E.	473		
Valiquette, Carolyn M.	491		
Wagner, Douglas	481		
Wang, Sheng-Jyh	1063		
Weaver, Glenn	1133		
Webb, Jon A.	1005		
Weems, Charles	39, 1133, 1145		
Wegesin, Dominic	481		
Weiss, Isaac	789		
Weiss, Richard	939		
Wells III, William M.	839		
Welty, R.	301		
Wheeler, Mark D.	811		

FOREWORD

Some DARPA IU activities for 1992 of interest to the IU community are given below.

DARPA Organization

In 1992 the program structure of the Software and Intelligent Systems Office (SISTO) was reorganized. As shown in Figure 1, IU projects are now in the Autonomous Systems portion of Intelligent Systems. (Demo-II is the Unmanned Ground Vehicle project.) The goals and missions of IU are unchanged.

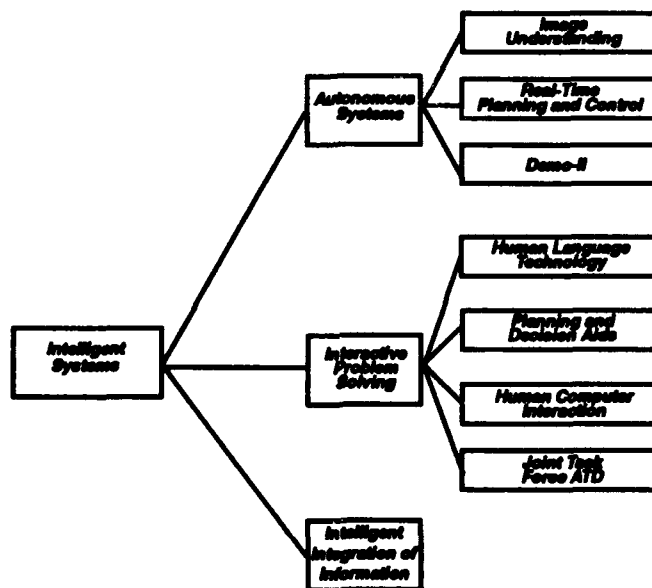


Figure 1. Program Structure of DARPA SISTO

Applied Technology Demonstration Support

At DARPA the model for insertion of research into the "real world" is to set up applied technology demonstrations (ATDs) such as RADIUS or the Unmanned Ground Vehicle (UGV) as practical demonstrations of the ultimate uses of research.

Unfortunately, by the end of a 3 to 5 year ATD the base technology is no longer on the cutting edge. Rand Waltzman, my predecessor, came up with a solution for RADIUS. He suggested a Broad Area Announcement (BAA) timed so as to activate a set of research projects a year after RADIUS was initiated to deal with perceived IU technology gaps. Proposed research projects are intended to supply current technology to RADIUS, but not to be on the critical path. Five such research projects in IU-RADIUS were initiated in 1992 and are reported on in these proceedings. We have used the same approach for the UGV project in the area of reconnaissance, surveillance, and target acquisition (RSTA). A BAA in six areas related to UGV RSTA was announced in December of 1992, and a set of awards will be made in early 1993. We hope that the results of these studies will transition into later UGV demonstrations. Some care must be exercised in implementing such studies:

- someone must coordinate and integrate the various research efforts with the ATD
- the original contract with the ATD contractor must include tasks for interacting with the researchers and for integrating the research results
- software environment standards for integrating the results of the research into the ATD must be part of the requirements of the research contracts
- the research must not be warped into development by the pressures of the ATD milestones

The major benefits of this approach are that real-world problems and associated imagery can be made available to the IU community and that advanced research can be incorporated into an ATD in advanced stages some time after the ATD has been initiated.

Special IU Workshops

Several special IU workshops were held in 1992. Profs. Ruzena Bajcsy (U Penn) and Takeo Kanade (CMU) hosted a computational sensors workshop at the U of Penn in May 1992. Profs. Ryszard Michalski (GMU) and Azriel Rosenfeld (UMd) hosted a workshop in learning in IU in October 1992. A session on benchmarking in IU was held at the Principal Investigators workshop in September 1992. Reports on these workshops appear in these proceedings.

IU/AI Efforts

Typically, IU researchers do not communicate with researchers in AI and vice versa. An attempt is being made to bring specialists in AI and IU together. Recent efforts include IU and learning (UMd/GMU), IU and reasoning (ISI/USC), IU and natural language (SUNY Buffalo), and IU and neural nets (new BAA; contracts to be awarded early 1993). Although the current efforts are small, it is hoped that they will lead to more extensive AI/IU interactions.

Automatic Target Recognition (ATR)

An interoffice DARPA working group on ATR has been set up to develop an interdisciplinary approach to ATR problems. The participants are Software and Intelligent Systems Technology Office (SISTO), Microelectronics Technology Office (MTO), Advanced Systems Technology Office (ASTO), and Defense Sciences Office (DSO). A joint BAA on ATR focussed on university participation was issued in late 1992; awards are expected by Spring of 1993.

Oscar Firschein, DARPA SISTO
Program Manager
Image Understanding

ACKNOWLEDGEMENTS

The twenty-second DARPA Image Understanding (IU) Workshop was held in Washington, D.C. on April 19-21, 1993. The workshop was sponsored by the Software and Intelligent Systems Technology Office of the Defense Advanced Research Projects Agency (DARPA). Prof. Larry Davis, University of Maryland (UMd), was the conference chairman. He was responsible for the organization of the technical sessions, and specifically for the special session on Young Investigators.

The first section of these proceedings contains research reviews presented by the principal investigators of the research institutions included in the DARPA program. The second section deals with current and future applications of IU. The third section contains technical papers prepared by members of the technical staffs of each institution. Although these technical papers were not presented at the workshop due to lack of time, they are included to present a complete record of recent results by the DARPA IU community.

The cover was prepared by Barbara Burnett of the Center for Automation Research (CFAR) at the University of Maryland to show several aspects of their IU research. The publisher is Morgan Kaufmann; DynCorp Meridian assisted in the production process.

The conference coordinator was Lois Hollan of PRC, Inc./DSR. She was a one-person team who handled the conference arrangements, and the thousands of large and small details that go into putting a conference together.

Section I

Principal Investigator Reports

Maryland Progress in Image Understanding

Yiannis Aloimonos

Larry S. Davis

Rama Chellappa

Azriel Rosenfeld

Computer Vision Laboratory, Center for Automation Research,
University of Maryland, College Park, MD 20742-3275

Abstract

Research in the Computer Vision Laboratory at Maryland is focused on both theoretical and normative questions related to vision. This report reviews our work on these questions during the period October 1991–January 1993. The areas covered include navigation, recognition, and low-level vision.

1 Introduction

Understanding the mechanisms underlying the processes of visual perception and creating machines with visual capabilities requires that we answer several questions of different natures. Among these are theoretical questions, whose answers will establish the range of possible mechanisms that *could* exist in intelligent visual systems; and normative questions, whose answers will suggest what classes of systems (animals or robots) would be desirable or optimal for a given set of tasks.

Our theoretical work on navigation is devoted to the analysis of correspondence and the investigation of the amount of three-dimensional information contained in noisy correspondence (or optical flow) fields; as well as to such issues as the analysis of localization techniques on natural terrain and the problem of visibility as it relates to path planning. Our research on normative questions related to navigation addresses the amount of information contained in normal flow fields that is necessary for robustly solving various specific problems, as opposed to problems of general recovery. Both aspects of our navigation-related research are reviewed in Section 2.

Our theoretical work on recognition has concentrated on the study of local projective and affine invariants, while our normative research on recognition has been devoted to the development of a framework for recognizing an object's purpose. Section 3 summarizes the main results of our recognition-related research.

We have also developed a collection of low-level vision techniques for image segmentation, segmentation of SAR data, and robust estimation, as well as new representations for objects that facilitate recognition tasks. Finally, we have worked in several specific application areas such as handwriting, face recognition, aerial image understanding, image enhancement and morphing, as well as on the parallelization of image understanding

algorithms. Our research on low-level vision, applications, and computational aspects is summarized in Section 4.

Appended to this report is a list of the 52 technical reports on computer vision issued by our Laboratory during the period October 1991–January 1993. The numbers in brackets in the body of this report refer to these technical reports.

2 Navigation

Visual navigation constitutes a problem which is of considerable practical as well as scientific interest. Navigation, in general, refers to the performance of sensory-mediated movement, and visual navigation is defined as the process of motion control based on an analysis of images. A system with navigational capabilities interacts adaptively with its environment. The movement of the system is governed by sensory feedback which allows it to adapt to variations in the environment; it does not have to be limited to a small set of predefined motions, as is the case for instance, with cam-activated machinery.

Visual navigation encompasses a wide range of perceptual capabilities that can be classified hierarchically. At the bottom of the hierarchy are low-level tasks, such as obstacle avoidance; the top is represented by high-level abilities like homing or target pursuit. As a basic capability, however, every visual navigation system must have an understanding of visual motion. It should be able to estimate the three-dimensional motions of objects in its environment; even more important, it should be able to determine its own motion. Naturally, a large part of our research is devoted to problems of visual motion analysis.

One way to deal with the problem of visual navigation is to consider it as a subproblem of the general structure from motion problem (a theoretical question). By making various assumptions we can develop solutions to the problem of token correspondence. In general, such solutions will involve errors, but we can study ways of identifying special instances of the problem in which a robust solution for structure and motion is possible. Our work along these lines is described in detail in Section 2.1. Section 2.2 is devoted to a normative study of the visual motion analysis problem, where we do not attempt to estimate feature correspondences; rather, as input to our motion algorithms we use the spatiotemporal derivatives

of the image intensity function (the so-called "normal flow").

2.1 Motion and structure estimation

2.1.1 Monocular and binocular recovery of motion and structure parameters

A central problem in vision-based navigation is to use 2-D information from a sequence of images to infer 3-D motion and structure information. By its very nature this problem is ill-posed and most of the algorithms discussed in the literature have proven to be very sensitive to even moderate levels of noise in the images and in the calibration of the camera(s).

Over the last few years, we have advocated the use of feature-based algorithms and long sequences of images for estimating the motion of the observer, the motions of objects, and the spatial structure of feature points. These efforts have resulted in several robust algorithms which have been successfully used for both monocular and binocular real image sequences.

In [41], the problem of estimating the kinematics of the moving camera and the spatial structure of the objects in a stationary environment is considered. Two estimation techniques, batch and recursive, have been used. The batch technique applies a non-linear least squares method to the stack of images, while the recursive technique uses an iterative extended Kalman filter and analyzes one frame at a time. The approach is based on modeling the motion of the camera using nine parameters, the 3-D coordinates of the rotation center and the linear and angular velocity components. A perspective camera model is used. The structure parameters are the 3-D coordinates of the feature points in the inertial coordinate system. These choices of parameters give rise to linear plant models, leading to closed form solutions for the state and covariance transition differential equations. Time consuming numerical integration steps are not needed.

The inputs to the algorithm are feature point correspondences over the image sequence. The task of automatically detecting and tracking features over a long sequence of consecutive frames is a challenging problem when the camera motion is significant. In general, feature displacement over consecutive frames can approximately be decomposed into two components: (i) the displacement due to camera motion, which can be compensated by image rotation, scaling, and translation; (ii) the displacement due to object motion and/or perspective projection. The displacement due to camera motion is usually much larger and more irregular than the displacement caused by object motion and perspective deformation. We have developed a two step approach: First, the motion of the camera is compensated using a recently developed image registration algorithm. Then consecutive frames are transformed to the same coordinate system and the feature correspondence problem is solved as one of tracking moving objects using a still camera. Methods of subpixel accuracy feature matching and tracking are introduced. The approach results in a robust and efficient algorithm. Results on several real image sequences are presented in two papers that appear elsewhere in

these Proceedings (parts of which have already been reported in [31, 41, 45]). The monocular algorithm has also been extended to the case of a binocular moving camera. For binocular imagery, the traditional stereo triangulation method fails when the images are not taken by the two cameras at the same time. But for our algorithm, since asynchronism is allowed, the two cameras can function independently (see [45]).

The methods summarized above have attempted to automate the problem of motion and structure recovery under relatively general conditions. In practical applications, such as the navigation of an autonomous vehicle or a low-flying aircraft, several simplifications are possible; for example, the 3-D structure of a (small) set of landmark points may be available from laser radar range measurements, or approximate vehicle kinematics may be known from inertial sensors. Batch and recursive estimation procedures for including such additional information from the sensors and the scene are described in [16]. For the situation where the structure of a set of landmark points is known, the absolute pose and velocity of the vehicle and the locations of the unknown feature points can be estimated. When the approximate vehicle kinematics are known, the ranges of the feature points and improved estimates of the vehicle kinematics can be obtained, as described in [16].

2.1.2 MAP estimation techniques [38, 47]

We have developed a Maximum A Posteriori (MAP) estimation algorithm for calculating the camera motion and the structure of a (rigid) scene. Our algorithm assumes the motion to be along a smooth trajectory and the sequence of images to be dense, so that the displacement between successive frames obtained by each camera is at most n pixels, where typically $n = 2$. We calculate instantaneous estimates of the focus of expansion (FOE) and of the scene depth map, and keep updating these estimates through the sequence. Our algorithm begins by calculating a MAP estimate of the subpixel displacement at each point and a confidence measure in that estimate. Using points for which the confidence is high we calculate MAP estimates for the FOE and the magnitudes of the displacements at these points, hence their relative depths. After determining the FOE we know the direction of displacement at every point in the image and we can again apply the MAP estimation method to get the displacement magnitude at each point and the associated confidence measure. This information is propagated over a long sequence of images by using the a posteriori distribution calculated from a set of images as a prior for the next set of images.

We have also developed a MAP algorithm for fusing monocular and stereo cues from two image sequences to get robust estimates of both motion and structure, under the same assumptions. The algorithm starts by calculating the instantaneous FOE, a MAP estimate of the displacement at each pixel, an associated confidence measure, and a relative depth map, as described above, from one of the two frame sequences. By calculating the disparities at some feature points and using information about their relative depths we compute the in-

stantaneous component of velocity in the direction perpendicular to the image plane. Using this information a depth map is calculated; this depth map is then used to derive a prior probability distribution for disparity that is used in matching the two frames of the stereo pairs. We use this method to estimate the disparity at each pixel independently; no assumptions about surface smoothness are used. Both the monocular and binocular algorithms have been successfully tested on real image sequences.

2.1.3 Frenet-Serret motion [36]

We have formulated a new model, *Frenet-Serret* motion, for the motion of an observer in a stationary environment. This model relates the motion parameters of the observer to the curvature and torsion of the path along which the observer moves. We derive screw-motion equations for Frenet-Serret motion and use them for geometrical analysis of the motion as well as analysis of the resulting velocity patterns in 3-D and motion field patterns on the surface of the velocity egosphere. We use normal flow to derive constraints on the rotational and translational velocity of the observer and compute egomotion by intersecting these constraints. We analyze the accuracy of egomotion estimation for different combinations of observer motion and feature distance. We suggest that depth of field should be controlled in order to make the analysis of egomotion on the basis of normal flow possible, and we derive the constraints on depth which make either rotation or translation dominant. These ideas have been validated by experiments on real image sequences.

2.1.4 Feature-based and flow-based motion estimation: a unified view [23]

State-of-the-art algorithms for computing 3-D motion from images can make use of either feature correspondences or optical flow. In particular, noise-robust algorithms can be formulated for the feature-based two-view problem—computing the depths of the feature points and the camera motion from correspondences of feature points between two images. For such algorithms, conditions for decomposability and for uniqueness of the solution, as well as direct optimization solutions and “critical surface” conditions, can be formulated. Similarly, noise robust algorithms can be formulated that make use of optical flow; here too, decomposability, uniqueness, direct optimization, and the “critical surface” can be treated, and relationships to the algorithms for finite motion can be analyzed. In both the feature-based and flow-based cases, a simpler treatment can be given for the case of motion on a planar surface.

2.2 Direct motion analysis

We have also addressed the problem of estimating 3-D motion directly without going through the intermediate stage of optical flow or correspondence estimation. The inputs that we have utilized are the spatiotemporal derivatives of the image intensity function (the normal flow).

From measurements on the image we can only compute the relative motion between the observer and any

point in the 3-D scene. The model that has usually been employed in previous research to relate 2-D image measurements to 3-D motion and structure is that of rigid motion. Consequently, egomotion recovery for an observer moving in a static world has been treated in the same way as the estimation of an object's 3-D motion relative to an observer. The rigid motion model is appropriate if only the observer is moving, but it holds only for a restricted subset of moving objects, mainly man-made ones. Indeed, virtually all objects in the natural world move non-rigidly. However, if we consider only a small patch in the image of a moving object, a rigid motion approximation is legitimate. For the case of egomotion, data from all parts of the image plane can be used, whereas for object motion only local information can be employed. We have therefore developed conceptually different techniques for explaining the mechanisms underlying the perceptual processes of egomotion recovery and 3-D object motion recovery.

We have developed solutions to the following problems: (a) *Given an active observer viewing an object moving in a rigid manner (translation + rotation), recover the direction of the 3-D translation and the time to collision by using only the spatiotemporal derivatives of the image intensity function.* Although this problem is not equivalent to “structure from motion” because it does not fully recover the 3-D motion, it is of importance in a variety of situations. If an object is rotating around itself and also translating in some direction, we are usually interested in its translation—for example, in problems related to tracking, prey catching, interception [27], obstacle avoidance, etc. The basic idea of this motion parameter estimation strategy lies in the employment of fixation and tracking [24, 46]. Fixation simplifies much of the computation by placing the object at the center of the visual field, and the main advantage of tracking is the accumulation of information over time. We have shown how tracking is accomplished using normal flow measurements, and have used it for two different tasks in the solution process: First, as a tool to compensate for the lack of existence of an optical flow field, and to estimate the translation parallel to the image plane; and second, to gather information about the motion component perpendicular to the image plane. (b) *Given an active observer moving rigidly in a static environment, recover the direction of its translation and its rotation.* This is the task of passive navigation, a term used to describe the set of processes by which a system can estimate its motion with respect to the environment. Our approach to egomotion estimation [32] is based on a geometric analysis of the properties of the normal flow field. The fact that the motion is rigid defines geometric relations between certain values of the spatiotemporal derivatives of the image intensity function. We have proved that the normal flow gives rise to global patterns in the image plane. The geometry of these patterns is related to the three dimensional motion parameters. By locating some of these patterns, which depend only on subsets of the motion parameters, using a simple search technique, the 3-D motion parameters can be found. The algorithmic procedure that we have developed (which is

described in a separate paper in these Proceedings) is provably robust, since it is not affected by small perturbations in the local image motion measurements. In fact, since only the signs of the normal flow measurements are employed, the direction of translation and the axis of rotation can be estimated in the presence of up to 100% error in the image measurements.

2.3 Localization, visibility, and path planning

2.3.1 Localization

We have developed an approach to autonomous localization of ground vehicles on natural terrain [4]. The localization problem is solved using measurements including altitude, heading, and distances to specific environmental points. Our algorithm utilizes random acquisition of distance measurements to prune the possible location(s) of the viewer. The approach is also applicable to airborne localization. The computational complexity of an implementation on the Connection Machine and the accuracy of the localization have been analyzed.

A method for localization and positioning in an indoor environment has also been developed [33]. We define localization as the act of recognizing the environment, and positioning as the act of computing the exact coordinates of a robot in the environment. Our method is based on representing the scene as a set of 2-D views and predicting the appearances of novel views by linear combinations of the model views. The method accurately approximates the appearance of scenes under weak perspective projection. Analysis of this projection as well as experimental results demonstrate that in many cases this approximation is sufficient to accurately describe the scene. When the weak perspective approximation is invalid, either a larger number of models can be acquired or an iterative solution to account for the perspective distortions can be employed. The method has several advantages over other approaches. It uses relatively rich representations; the representations are 2-D rather than 3-D; and localization can be done from only a single 2-D view. The same general method is applied to both the localization and positioning problems, and a simple algorithm for repositioning, the task of returning to a previously visited position defined by a single view, can be derived from this method.

2.3.2 Visibility and path planning

We have investigated [29] two classes of parallel algorithms for point-to-region visibility analysis on terrain: ray-structure-based methods and propagation-based methods. A new propagation-based algorithm has been developed which avoids problems commonly occurring with such algorithms. The performance and characteristics of the two kinds of algorithms have been compared. The sources of uncertainty in visibility computation and the importance of taking uncertainty into consideration have been analyzed. Different methods for representing the uncertainty have been studied, including Monte Carlo simulation, analytic estimation, and some simple heuristic indicators. Our experiments show that these indicators can be used for efficient coarse classification of the likelihood of point intervisibility.

Current approaches to robot motion planning are limited in their ability to deal with an uncertain and dynamically changing environment. We have developed [5] a probabilistic model based on discrete events that abstract the dynamic interaction between the robot and the unknown part of the environment. The resulting framework makes it possible to design and evaluate motion planning strategies that consider both the known portion of the environment and the portion that is unknown but satisfies a probability distribution. We have studied three instances of the general model that have been useful in designing efficient motion planning algorithms under various assumptions about the robot's environment and its behavior with respect to unexpected events.

3 Recognition

The problem of object recognition has been traditionally treated as one of matching image features or recovered surface features with geometric object models. Such approaches are primarily devoted to the robust detection or recovery of features and to handling the combinatorial complexity of the matching process. In this spirit, the problem of recognition is defined as finding regularity across views, and the theories of object recognition can be classified into three main groups: computation of invariant properties, object decomposition into parts, and alignment. In Section 3.2 our recent work on invariants is presented, with emphasis on local projective and affine invariants. Section 3.3 is devoted to a novel method of two-dimensional object segmentation and recognition, and Section 3.4 deals with our recent work on alignment (pose estimation). Section 3.1 describes our recent work on an alternative framework for recognition.

3.1 A framework for object recognition [10]

Vision systems that operate in different environments and perform different visual tasks do not necessarily recognize objects using similar algorithms. A vision system that needs to recognize ten types of objects does not necessarily work in the same way as a system that needs to recognize one type or a hundred types. A system that serves a rapidly moving agent is not necessarily built in the same way as a system for a stationary agent. Object recognition should be studied by taking into account not only the objects that have to be recognized but also the agent that has to perform the recognition. Since different agents, working with different purposes in different environments, do not recognize visually in the same manner, we should not seek a general, universal theory of object recognition. Instead, we should concentrate on developing a methodology that, given an agent in an environment, will suggest how to perform particular recognition tasks.

An agent is a robot that has visual (and other) sensing capabilities and is able to carry out a set of behaviors. These behaviors are direct results of a set of purposes or intentions that the agent has. A behavior is identified as anything that changes the internal state of the agent and its relationship to the environment. Carrying out a behavior calls for the performance of various recognition

tasks. By performing partial recovery of attributes of an object, we can find out if the object is suitable for the desired purpose. In general an object can be used for many purposes. The agent must recognize the one needed to carry out its behavior.

Perception is a causal and intensional transaction between the mind and the world. The intensional content of our visual perception is termed "the visual experience". When we see a table there are two elements in the perceptual situation: the visual experience and the table. The two are not independent. The visual experience has the presence and features of the table as conditions of satisfaction. The content of the visual experience is self-referential in the sense that it requires that the state of affairs in the world must cause the visual experience which is the realization of the intensional content.

When we visually perceive an object we have a visual experience. This visual experience is an experience of the object. It may be that the conditions of satisfaction are not fulfilled. This is the case for illusions, hallucinations, etc. The visual experience, and not the world, is at fault. The visual experience that we have, in this case, is indistinguishable from the visual experience we would have if we actually saw the real object. The intensional content of the visual experience determines its conditions of satisfaction. A visual experience in that sense is a mental phenomenon which is intrinsically intensional.

An agent is defined as a set of intentions, I_1, I_2, \dots, I_n . Each intention I_k is translated into a set of behaviors, $B_{k1}, B_{k2}, \dots, B_{km}$. Each behavior B_{ki} calls for the completion of recognition tasks $T_{ki1}, T_{ki2}, \dots, T_{kij}$. The agent acts in behavior B_{ki} under intention I_k . The behavior calls for the completion of recognition tasks T_{ki1}, \dots, T_{kin} . The behavior sets parameters for the recognition tasks. Under one behavior a chair will answer yes to a recognition task that is looking for obstacles, under another behavior it will answer yes to a task that is looking for a sitting place, and under still another it will answer yes to a task that is looking for an assault weapon.

We view the recognition process along the axis (intention, behavior, recognition task). For a theory of purposive object recognition we should be able to make two basic transformations: first, from a desired intention to the set of behaviors that achieve it; second, from a specific behavior to some needed recognition task(s). We have shown [10] that the intention-to-behaviors problem with a finite number of behaviors is undecidable by reducing it the halting problem. We believe that the transformation from behaviors to recognition tasks is also hard.

If we add constraints to our definition of the problem we can move from undecidability to intractability. For example, by constraining ourselves to a constant set of objects we can show a PSPACE-hard lower bound. This can be shown by reducing our problem, for example, to that of motion planning for an object in the presence of movable obstacles, where the final positions of the obstacles are specified as part of the goal of the motion. The reduction is straightforward. The set of objects contains the moving objects and the obstacles. The positions of

the objects are part of the relation set. The intention encodes the final state. Grasping, pushing and moving are the behaviors. Solving the intention-to-behaviors problem gives a solution to this problem.

We are interested in object utilization; this is not the same as naming an object. Under our framework an agent acts in behavior B_{ki} under intention I_k . The behavior calls for the completion of recognition tasks T_{ki1}, \dots, T_{kin} . The behavior sets parameters for the recognition tasks. Each recognition task activates a different collection of basic perceptual modules. Each module qualitatively finds a generic object property which is a result of one or a combination of direct low-level computations on some sensory data (possibly done by other modules). The result of a module's operation is given as a qualitative value. Each module has its own neighboring open intervals which are parameter-specific. The i^{th} module can take one of q_{i1}, \dots, q_{in} qualitative values.

The state of our recognition system, denoted by Q_i , is a tuple of all the qualitative values of our modules (q_1, \dots, q_m) under recognition task T_{kij} . Each recognition task T_{kij} defines a system state that will constitute a positive answer to that recognition task. Recognition is done when we complete our task, which means a stable answer from our modules. The conditions for this kind of decision will not be considered here and probably should take into account utility measures (frequency of appearance, network complexity, etc.).

Under this framework learning can be defined as the process of matching the "correct" system state with the recognition task needed by a certain behavior. This process is actually the reverse of recognition. A behavior creates a need for an object. An object is segmented by low level modules, and a system state is achieved. The object is tested and a satisfied result for a needed behavior starts the creation or definition of a recognition task.

When we need to perform a given recognition task T_{kij} under behavior B_{ki} and intention I_k , we may assume that some parameter setting is done by the intention and the behavior. These parameters fix the setting for the task, which includes the required system state (some of the modules might be in don't care states) and possibly some additional "common knowledge" parameters, such as environmental parameters (outdoor, indoor), predator, size, etc. From this point of view the recognition process makes use of high-level information. For further discussion of object categories and functional modeling see [10].

3.2 Invariants [19, 34, 44]

Invariants are useful in solving major problems associated with object recognition. For instance, different images of the same object often differ from each other because of the different viewpoints from which they were taken. To match the two images, standard methods thus need to find the correct viewpoint, a difficult problem that can involve search in a large parameter space of all possible points of view and/or finding feature correspondences. Geometric invariants are shape descriptors, computed from the geometry of the shape, that remain un-

changed under geometric transformations such as change of viewpoint. Thus they can be matched without search. Deformations of objects are another important class of changes for which invariance is useful.

We have developed a new and more robust method of obtaining local projective and affine invariants. These shape descriptors are useful for object recognition because they eliminate the search for the unknown viewpoint. Being local, our invariants are much less sensitive to occlusion than the global ones used by others. The basic ideas underlying our method are: i) employing an implicit curve representation without a curve parameter, thus increasing robustness; ii) using a canonical coordinate system which is defined by intrinsic properties of the shape, independently of any given coordinate system, and is thus invariant. Several shape configurations have been treated using this approach: a general curve without any correspondence, and curves with known correspondences of one or two feature points or lines. The method is applied by fitting an implicit polynomial in a neighborhood of each object contour point. It has been successfully implemented for real images of various two-dimensional objects in three-dimensional space. This work is described in detail in a separate paper in these Proceedings.

3.3 Target recognition [52]

A multilevel energy environment has been developed that simultaneously performs delineation, representation and classification of two-dimensional object shapes in an image utilizing a global optimization technique. The energy environment supports a novel multipolar representation which allows the delineation and representation tasks to be viewed as a single operation. The delineator acts as a hypothesis generator for the multipolar representation, which uses description length tests to determine whether to establish new "centers". Model information is then utilized at these centers to identify pieces of objects. In this way occluded objects can be recognized. This method is more robust than conventional, multistaged approaches because it incorporates all known information into a single decision process. It has been applied to the delineation and classification of vehicles in FLIR images. Further details on this work can be found in a separate paper in these Proceedings.

3.4 Pose estimation

We have shown that the bounded error recognition problem for images of non-planar three-dimensional objects using point features can be decomposed into a set of one-dimensional search tasks, involving searches along lines joining the origin of the object coordinate system to the feature points chosen to model the object. Points are selected along these lines at locations given by the coordinates of the detected image points; concurrent bracketing of these points by segment tree search along each line provides maximal matchings between feature points and image points. The depth of search is limited by pixel resolution. This method is well adapted to the task of tracking objects in the presence of variable occlusion and clutter. This work is described in greater detail in a sep-

arate paper in these Proceedings. Some of our earlier work on object pose estimation is described in [12].

4 Low-level vision

4.1 Estimation and segmentation

4.1.1 Image segmentation

The problems of image estimation and segmentation can be cast in a joint Maximum A Posteriori (MAP) framework using Gibbs distributions defined over the image intensities and over line processes representing the boundaries of image regions. MAP estimation is then reduced to minimizing an appropriate energy function defined on the intensity and line processes.

The energy function typically has three components; (a) a measure of closeness to the data, (b) a weak constraint which assumes that the image is mostly smooth except at the discontinuities, and (c) penalties on broken contours, multiple edges, etc. In its most general form, the energy is highly non-convex, causing deterministic relaxation techniques to converge to shallow, local minima. Stochastic relaxation is not always a viable alternative due to the computational complexity of the problem. We are interested in deterministic, continuation methods to solve the problem.

Alternative energy functions have been suggested which depend primarily on the intensities and usually ignore the interactions between the line processes. We can utilize the insights gained by these methods by showing that each of the alternative energy function sequences has an equivalent sequence in the domain of the intensity and line processes. Interactions can then be added once the equivalent energy functions have been obtained. There are many equivalent energy functions in the domain of the intensity and line processes; the concept of an uncertainty function can help us to choose the proper equivalent energy function. The uncertainty function is analogous to the entropy in a statistical mechanical system.

The resulting algorithm is a combination of the Conjugate Gradient and the Iterated Conditional Models algorithms and is completely deterministic. It has been applied successfully to the segmentation of aerial images [1].

A segmentation-based image coding technique has also been developed [17]. Both uniform and textured region extraction algorithms are used for segmentation. Textured regions are reconstructed using 2-D noncausal Gaussian-Markov random field models. Uniform regions are reconstructed using polynomial expansions. An arithmetic coder is used for coding the boundaries of regions. Reasonable quality images have been obtained at a compression factor of 82:1.

4.1.2 Segmentation of SAR data [18, 40]

A statistical image model has been developed for segmenting polarimetric synthetic aperture radar (SAR) data into regions of homogeneous and similar polarimetric backscatter characteristics. A model for the conditional distribution of the polarimetric complex data is combined with a Markov random field representation for

the distribution of the region labels to obtain the posterior distribution. Optimal region labeling of the data is then defined as maximizing the posterior distribution of the region labels given the polarimetric SAR complex data. This MAP technique has been implemented on a parallel optimization network. Two procedures can be used for selecting the characteristics of the regions; one is supervised and requires training areas, the other is unsupervised and is based on multidimensional clustering of the logarithms of the parameters composing the polarimetric covariance matrix of the data. Experiments using real multilook polarimetric SAR complex data, dual polarization SAR data, and fully polarimetric SAR data indicate that all three types of data yield generally similar segmentation results.

For unsupervised segmentation, classes of polarimetric backscatter have been selected based on multidimensional fuzzy clustering. The clustering procedure uses both polarimetric amplitude and phase information, is adapted to the presence of image speckle, and does not require an arbitrary weighting of the different polarimetric channels; it also provides a partitioning of each data sample used for clustering into multiple clusters. Given the classes, the entire image can then be classified using a MAP polarimetric classifier. Successful segmentation results have been obtained using four-look polarimetric SAR complex data of lava flows and of sea-ice acquired by the NASA/JPL airborne polarimetric radar (AIR-SAR).

4.1.3 Robust estimation [30]

Data processing for scientific and industrial tasks often involves accurate extraction of theoretical model parameters from empirical data, and requires automated estimation methods that are robust in the presence of "noisy" (i.e., contaminated) data. Robust estimation is thus an important statistical tool that is frequently applied in numerous fields of science and engineering.

Since the computational complexity of an estimator is one of the most important measures of its practicality, searching for methods that reduce the time (and space) complexity of robust estimators is a desirable research goal. We have developed several computationally efficient algorithms for the exact computation of robust statistical estimators. In particular, we have studied the design and analysis of such algorithms for various problem domains, including line, curve, and surface fitting. In this connection, we have developed a general methodology for the efficient computation of these classes of estimators through the application of computational geometry techniques. In particular, we have developed randomized algorithms for the above tasks that have the following properties: (1) The algorithms always terminate and return the correct computational results; (2) the improved (expected) running times occur with extremely high probability; (3) the algorithms are quite easy to implement; (4) the constants of proportionality (hidden by the asymptotic notation) are small (i.e., the algorithms are practical); and (5) the algorithms are space optimal (i.e., they require linear storage).

The problem of fitting a straight line to a set of data

points is an important task in many application areas. Recently the computation of linear estimators that are robust has been recognized as important, since these estimators are insensitive to outlying data points, which arise often in practice. We have studied one such robust estimator [13], Siegel's repeated median (RM) line estimator, which achieves the highest possible breakdown point of 50%, and have developed: (1) a simple practical randomized RM algorithm that runs in $O(n \log^2 n)$ time with high probability, and (2) a slightly more complex randomized RM algorithm which performs as well asymptotically, and which is shown by empirical evidence to perform in time $O(n \log n)$ on many realistic input distributions.

Existing algorithms for affine equivariant regression estimators with high breakdown point are computationally intensive. Heuristically, this appears to be due to combinatorial and geometric reasons. Consequently, non-affine estimators may allow faster computation. We have developed [43] an RM algorithm which runs in $O(n \log^2 n)$ time, a substantial improvement over the naive $O(n^2)$ method. The new algorithm allows an empirical study of this estimator for n up to 40,000. It turns out that the finite-sample efficiencies converge extremely slowly although the estimator is asymptotically normal.

More generally, given a set of n distinct points in d -dimensional space that are hypothesized to lie on a hyperplane, robust statistical estimators have been recently proposed for the parameters of the model that best fits these points. We have developed [48] efficient algorithms for computing median-based robust estimators (e.g., the RM and Theil-Sen estimators) in high-dimensional space, by generalizing the computational geometry techniques that were used to achieve efficient algorithms in the 2-D case. This yields $O(n^{d-1} \log n)$ expected time algorithms for the d -dimensional Theil-Sen and RM estimators. Both algorithms are space optimal, i.e., they require $O(n)$ storage, for fixed d . An extension of the methodology to nonlinear domain(s) such as circle fitting has also been demonstrated.

4.1.4 Reliability of geometric computations [22]

The reliability of 3-D interpretations computed from images can be analyzed in statistical terms by employing a realistic model of image noise. First, the reliability of edge fitting can be evaluated in terms of image noise characteristics. Then, the reliability of vanishing point estimation can be deduced from the reliability of edge fitting. The result can then be applied to focal length calibration, and an optimal scheme derived in such a way that the reliability of the computed estimate is maximized. The confidence interval of the optimal estimate can also be computed. We can also evaluate the reliability of fitting an orthogonal frame to three orientations obtained by sensing. Finally, we can derive statistical criteria for testing edge groupings, vanishing points, focuses of expansion, and vanishing lines.

4.2 Representation and geometry

4.2.1 Multipolar representation [3]

We have developed a new method of delineating and representing lobed objects, i.e., objects containing multiple compact parts, by using a multi-polar representation (MPR). The process begins by constructing a single-center polar representation somewhere inside the object. This establishes a 1-D, cyclic Markov Random Field (1DCMRF) which optimizes edge sharpness and contour smoothness. The optimization is done using simulated annealing. The 1DCMRF is segmented into sectors at significant minima of the radius; these sectors define lobes. (An alternative is to segment at zero-crossings of the contour curvature; see [52].) Next, each lobe is assigned a candidate polar sector representation centered at the lobe's centroid. This new set of representations is then compared to the previous one using a "radius entropy test" (RET). This test selects the representation with the highest degree of roundness, i.e. the representation in which the radii are most uniform in their sizes. When a new representation supersedes an old one, each new polar center establishes a 1-D Markov Random Field for its sector and boundary conditions are determined between neighboring MRFs. This process continues recursively within each of these MRFs. To deal with deep lobes and concavities we define two special classes of MRFs. Tunnel MRFs are used to explore long narrow lobes; these MRFs extend the original lobe center by dividing the lobe's radii into fore and aft groups. External MRFs, whose centers lie outside of the object, are used to delineate concavities. These are detected when significant contour gaps appear between neighboring MRFs. These MRFs must also meet the RET criterion in their creation. The method can operate on raw image data without preprocessing; the object representation is constructed simultaneously with the delineation process, an important advantage when using the representation for object recognition in real images, as described in [52].

4.2.2 Multiresolution curve representation [25]

We have developed a robust method for describing planar curves at multiple resolution using curvature information. The method takes into account the discrete nature of digital images as well as the discrete aspect of a multi-resolution structure (pyramid). The robustness of the technique is due to information that is extracted by observing the behavior of corners in the pyramid. The algorithm is conceptually simple and easily parallelizable. We have analyzed the curvature of continuous curves in scale-space, and studied the behavior of curvature extrema under varying scale. These results are used to eliminate any ambiguities that might arise from sampling problems due to the discreteness of the representation.

4.2.3 Growth models for shapes [51]

We have developed discrete models for growth of a shape from a point on a two-dimensional Cartesian grid. By growth is meant an accretionary process occurring at the boundary of the shape. We have studied three types of growth models: deterministic (periodic), probabilis-

tic (stochastic), and probabilistic mixing of deterministic processes. We have found that the probabilistic models can produce smooth isotropic or elongated shapes, concavities, and protrusions.

4.2.4 Polygonal ribbons [7, 20]

A polygonal ribbon is a finite sequence of polygons such that each pair of successive polygons intersect exactly in a common side. We have investigated various geometric properties of such ribbons in two and three dimensions, including properties such as nonselfintersection, orientability, and twist. When the polygons are all of simple types—for example, when they are all triangles or all rectangles—they can be represented compactly in terms of such quantities as vertex coordinates, side lengths, and dihedral angles. For nonselfintersecting ribbons, we have established basic connectivity properties of the ribbon and its border.

4.2.5 Connectedness [6]

In collaboration with Prof. A. Nakamura of Japan, we have solved the long-outstanding problem of proving that two-dimensional connected pictures over $\{0,1\}$ are not recognizable by finite-state acceptors.

4.3 Applications

4.3.1 Aerial image understanding

The University of Maryland (with TASC a subcontractor) is one of the group of institutions doing research on aerial image understanding in support of the RADIUS program. The emphasis of our research is on knowledge based change detection using site models and image analysts' (IA) domain expertise. Typically, an IA uses a set of object and background models to build a site model for an area of interest. Change detection consists of classifying changes in the imagery into changes due to site updates, changes due to activity, and irrelevant changes due to illumination, seasonal variations, etc.

Before change detection can be attempted, the newly acquired images have to be registered to the site model. Site models can also be used to mediate registration between two severely off-nadir images. We are developing methods of using site models in image registration; this is especially important in oblique acquisition situations where 3-D information is critical due to foreshortening, etc.

Even if the task of image registration can be accomplished, the IA's expertise is crucial in identifying relevant changes. This expertise is dependent on the site and the specific intelligence agenda. We are studying how image understanding (IU) techniques can aid the IA in change detection. This will be accomplished by designing an interface that allows the IA to specify what are to be considered as relevant changes, and to select appropriate IU algorithms for detecting these changes.

Once changes have been identified, updating of the site models may be necessary. We plan to use non-monotonic reasoning based techniques such as assumption based truth maintenance systems (ATMS) and their variants to efficiently perform the searches required for image registration; to formulate constraints on IU algorithms (e.g.,

for stereo or building detection); and to provide interactive user guidance in change detection.

A significant part of our research effort will be the employment of usability analysis techniques and guidance from experienced IAs to ensure the utility of the algorithms that we develop. A more detailed paper reporting our progress to date can be found elsewhere in these Proceedings.

4.3.2 Trees [11, 39]

Plants such as trees can be modeled by three-dimensional hierarchical branching structures. If these structures are sufficiently sparse, so that self-occlusion is relatively minor, we have shown that their geometrical properties can be recovered from a single image.

The distribution of leaves in a tree crown can be modeled by a random geometric process. Statistical properties of such distributions can then be derived, including the probability of seeing through the leaves, and the distribution of leaf gray levels under various illumination models.

4.3.3 Faces [15, 42]

Faces represent one of the most common visual patterns in our environment, and humans have a remarkable ability to recognize them. Face recognition does not fit into the traditional approaches of model based recognition in vision. Like most other natural objects, a geometrical interpretation of faces is difficult to achieve. We have developed a feature based approach to face recognition, where the features are derived from the intensity data without assuming any knowledge of the face structure. The feature extraction model is biologically motivated, and the locations of the features often correspond to salient facial features such as the eyes, nose, etc. Topological graphs are used to represent relations between features, and a simple deterministic graph matching scheme which exploits the basic structure is used to recognize familiar faces from a database. Each of the stages in the system can be fully implemented in parallel to achieve real time recognition.

We have also developed an approach to labeling the components of faces from range images. The components of interest are those which humans usually find significant for recognition. To cope with the non-rigidity of faces, an entirely qualitative approach is used. A pre-processing stage employs a multi-stage diffusion process to identify convexity and concavity points. These points are grouped into components and qualitative reasoning about possible interpretations of the components is performed. Consistency of hypothesized interpretations is verified using context-based reasoning.

4.3.4 Handwriting [8, 21]

The primary intention of the handwriting process is to produce a series of perceptually meaningful strokes which collectively relay a message to the reader. Unfortunately, the process may be quite complex, so noise is easily introduced and correct interpretation may not be possible in the absence of contextual knowledge (linguistic or graphic) about the domain. We believe that

a handwritten document should be analyzed within the context of the process which created it.

The problem of off-line handwritten character recognition has eluded a satisfactory solution for several decades. Researchers working in the area of on-line recognition have had greater success, but the possibility of extracting on-line information from static images has not been fully explored. The experience of forensic document examiners assures us that in many cases, such information can be successfully recovered.

We have designed a system for the recovery of temporal information from static handwritten images, based on local, regional and global temporal clues which are often found in hand-written samples, and have shown how these clues can be recovered from an image. Our approach attempts to understand the handwriting signal and to perform a detailed analysis of stroke and sub-stroke properties. We believe that the recovery task requires that we break away from traditional thresholding and thinning techniques, and we have developed a framework for such analysis. We have shown how temporal clues can reliably be extracted from this framework and have developed a control structure for integrating the partial information. Many seemingly ambiguous situations can be resolved by the derived clues and by knowledge about the writing process.

If we view handwriting as a parameterized process, then problems such as signature verification can be posed as the recovery of specific parameters. To demonstrate this approach, we have studied the mechanical aspects of instrument grasp and have qualitatively demonstrated the recovery of parameters which have stable and meaningful effects on the static image. Our model for the grasping of a writing instrument makes explicit the forces exerted in the hand/instrument/paper system while the instrument is in motion. We have used this model as a basis for analyzing the pressure exerted on the writing surface for strokes of different orientations. We have shown that relative pressure information is preserved in static handwriting images. This information is a valuable heuristic in recovering the direction of motion of the instrument which created the stroke, and can be applied to the development of on-line recognition techniques that can be used on off-line handwritten data.

4.4 Parallel processing

Single instruction stream, single data stream (SIMD) processor array machines are popular in practical parallel computing. Such machines differ from one another considerably in the level of autonomy provided to each processing element (PE) of the array. An understanding of the levels of autonomy provided by the architectures is important in the design of efficient algorithms for them. We can classify SIMD architectures into six categories differing in key aspects such as the selection of the instructions to be executed, operands for the instructions, and the source/destination of communications.

The data parallel model of computation used in processor arrays exploits the parallelism in the data by processing multiple data elements (pixels, in image analysis) simultaneously by assigning one PE to each data ele-

ment. This scheme does not make efficient use of the processor array when processing relatively small data structures. We have developed [37] a technique of data replication that combines operation parallelism with data parallelism, to process small data structures efficiently on large processor arrays. It decomposes the main operation into suboperations that are performed simultaneously on separate copies of the data structure. The autonomy of the individual PEs is critical to this decomposition. We have developed replicated data algorithms for several low level image operations such as histogramming, convolution, and rank order filtering [2]. Additionally, we have developed a way of constructing a replicated data algorithm for an operation automatically from an image algebra expression for it, demonstrating its generality. We have also devised a replicated data algorithm to compute single source shortest paths on general graphs, demonstrating its applicability beyond image analysis. We have analyzed the speedup performance of the algorithms on various interconnection networks to determine the conditions under which the technique results in a speedup. Implementations of the algorithms on a Connection Machine CM-2 and a MasPar MP-1 have yielded impressive speedups.

We have also developed [37] a parallel search scheme for the model-based interpretation of aerial images under a focus-of-attention paradigm and have implemented it on a CM-2. Candidate objects are generated as connected combinations of the connected components of the image and are matched against the model by checking if the parameters computed from the region satisfy the model constraints. This process is posed as a search in the space of combinations of connected components with the finding of an (optimally) successful region as the goal. Our implementation exploits parallelism at multiple levels by parallelizing control tasks such as the management of the open list. The level of processor autonomy and other details of the architecture play important roles in the search scheme.

We have defined [9] a class of routing operations which can be performed in n unit-routes on a hypercube with 2^n nodes. Specifically, we have shown that the conjunction of two conditions, called wrap-contiguity and wrap-mononicity, is sufficient to allow efficient routability. This result subsumes some earlier results on hypercube routing.

The use of dynamic programming for stereo matching has been studied extensively. It has been pointed out that this approach is suitable for parallel processing, but there have been no attempts to implement a dynamic programming stereo matching algorithm on a parallel machine. We have developed [26] a massively parallel implementation of Baker's dynamic programming stereo algorithm; our implementation can use many processors per scanline, compared to a naive approach of one processor per scanline. This is important because typical images contain 256 to 1024 scanlines, while massively parallel machines can have many more processors. We have also introduced a method of handling inter-scanline inconsistencies that is very well suited for parallel implementation. The method increases the amount of process-

ing needed to solve the stereo matching problem by only a small fraction. On a 16K processor Connection Machine the entire algorithm requires as little as 1 second for simple 512×512 images.

4.5 Miscellaneous

4.5.1 Enhancement [35] and morphing [50]

We are studying the use of a multi-stage physical diffusion process in early vision processing of range images. The input range data is interpreted as occupying a volume in 3-D space. Each diffusion stage simulates the process of diffusing the boundary of the volume into the volume. The result appears to be useful for both discontinuity detection and segmentation into shape coherent regions.

Image interpolation and metamorphosis can also be performed by using a scale space created by diffusing the difference function of the source and the goal images. This formulation allows us to minimize the need for human intervention in the selection of features in a process such as image metamorphosis. The smooth transitions are accompanied by a moderated blurring that is useful in displaying the metamorphosis process. The approach can also be applied to motion image sequences as a method of enhancing animation.

4.5.2 Matching [28]

In its original form the point pattern-matching relaxation scheme of Ranade and Rosenfeld did not easily permit the representation of uncertainty, and it did not exhibit the desirable property that confidence in consistent pairings of features should increase from one iteration to the next. Because the process of pooling intrinsic support with contextual support is essentially a process of evidence combination, it was suggested by Faugeras over ten years ago that the evidence theory of Dempster and Shafer might be an appropriate framework for relaxation labeling. We have implemented such an approach and applied it to the domain of object recognition in simulated SAR imagery.

4.5.3 Bibliographies [14, 49]

Bibliographies containing a total of over 3000 references on computer vision, image analysis, and related topics have been prepared for the years 1991 and 1992.

References

- [1] A. Rangarajan and R. Chellappa, "A Continuation Method for Image Estimation and Segmentation", CAR-TR-586, F49620-88-C-0067 and MIP-84-51010, October 1991.
- [2] P.J. Narayanan and L.S. Davis, "Rank Order Filtering on Processor Array Machines", CAR-TR-587, DACA76-88-C-0008, October 1991.
- [3] N.S. Friedland and A. Rosenfeld, "Lobed Object Delineation Using a Multipolar Representation", CAR-TR-590, AFOSR-91-0239, October 1991.
- [4] Y. Yacoob and L.S. Davis, "Computational Ground and Airborne Localization Over Rough Ter-

- rain", CAR-TR-591, DACA76-89-C-0019, November 1991.
- [5] R. Sharma, "A Probabilistic Framework for Dynamic Motion Planning in Partially Known Environments", CAR-TR-592, DACA76-89-C-0019 and IRI-90-57934, October 1991.
 - [6] A. Nakamura and A. Rosenfeld, "Two-Dimensional Connected Pictures Are Not Recognizable By Finite-State Acceptors", CAR-TR-593, November 1991.
 - [7] P. Bhattacharya and A. Rosenfeld, "Triangulated Ribbons in Two and Three Dimensions", CAR-TR-594, AFOSR-91-0239, November 1991.
 - [8] D.S. Doermann and A. Rosenfeld, "Recovery of Temporal Information from Static Images of Handwriting", CAR-TR-595, December 1991.
 - [9] T. Bestul, "A Class of Efficiently Routable Hypercube Operations", CAR-TR-596, December 1991.
 - [10] E. Rivlin, Y. Aloimonos, and A. Rosenfeld, "Purposeful Recognition: A Framework", CAR-TR-597, DACA76-89-C-0019 and IRI-90-57934, December 1991.
 - [11] A. Waksman and A. Rosenfeld, "Sparse, Opaque Three-Dimensional Texture, 1: Arborescent Patterns", CAR-TR-598, DACA76-89-C-0019, December 1991.
 - [12] D.F. DeMenthon and L.S. Davis, "Model-Based Object Pose in 25 Lines of Code", CAR-TR-599, DACA76-89-C-0019, December 1991.
 - [13] D.M. Mount and N.S. Netanyahu, "Computationally Efficient Algorithms for a Highly Robust Line Estimator", CAR-TR-600, DACA76-89-C-0019 and CCR-89-08901, December 1991.
 - [14] A. Rosenfeld, "Image Analysis and Computer Vision: 1991", CAR-TR-601, AFOSR-91-0239, January 1992.
 - [15] B.S. Manjunath, R. Chellappa, and C. von der Malsburg, "A Feature Based Approach to Face Recognition", CAR-TR-604, AFOSR-90-0133, January 1992.
 - [16] C. Shekhar and R. Chellappa, "Visual Motion Analysis for Autonomous Navigation", CAR-TR-607, N00014-89-J-1598, March 1992.
 - [17] O.-J. Kwon and R. Chellappa, "Segmentation-Based Image Compression", CAR-TR-608, MIP-91-00655, March 1992.
 - [18] E. Rignot and R. Chellappa, "Segmentation of Polarimetric Synthetic Aperture Radar Data", CAR-TR-609, F49620-92-J-0130, March 1992.
 - [19] I. Weiss, "Local Projective and Affine Invariants", CAR-TR-612, DACA76-92-C-0009, April 1992.
 - [20] P. Bhattacharya and A. Rosenfeld, "Rectangular Ribbons in Two and Three Dimensions", CAR-TR-613, AFOSR-91-0239, April 1992.
 - [21] D.S. Doermann and V. Varma, "Instrument Grasp: A Model and its Effects on Handwritten Strokes", CAR-TR-614, April 1992.
 - [22] K. Kanatani, "Statistical Reliability of 3-D Interpretation from Images", CAR-TR-615, DACA76-92-C-0009, April 1992.
 - [23] K. Kanatani, "Computation of 3-D Motion from Images", CAR-TR-617, DACA76-92-C-0009, April 1992.
 - [24] C. Fermüller and Y. Aloimonos, "Tracking Facilitates 3-D Motion Estimation", CAR-TR-618, IRI-90-57934, April 1992.
 - [25] C. Fermüller and W. Kropatsch, "Multi-Resolution Shape Description by Corners", CAR-TR-619, IRI-90-57934, April 1992.
 - [26] L.T. Chen and L.S. Davis, "Parallel Stereo Matching Using Dynamic Programming", CAR-TR-620, DACA76-92-C-0009, April 1992.
 - [27] L. Huang and Y. Aloimonos, "The Geometry of Visual Interception", CAR-TR-622, DACA76-92-C-0009 and IRI-90-57934, April 1992.
 - [28] P. Cucka and A. Rosenfeld, "Evidence-Based Pattern-Matching Relaxation", CAR-TR-623, AFOSR-91-0239, May 1992.
 - [29] Y.A. Teng and L.S. Davis, "Visibility Analysis on Digital Terrain Models and Its Parallel Implementation", CAR-TR-625, DACA76-92-C-0009, May 1992.
 - [30] N.S. Netanyahu, "Computationally Efficient Algorithms for Robust Estimation", CS-TR-1898, DACA76-89-C-0019 and AFOSR-91-0239, May 1992.
 - [31] Q. Zheng and R. Chellappa, "Automatic Feature Point Extraction and Tracking in Image Sequences for Arbitrary Camera Motion", CAR-TR-628, DACA76-92-C-0009, June 1992.
 - [32] C. Fermüller and Y. Aloimonos, "Qualitative Egomotion", CAR-TR-629, DACA76-92-C-0009 and IRI-90-57934, June 1992.
 - [33] E. Rivlin and R. Basri, "Localization and Positioning Using Combinations of Model Views", CAR-TR-631, DACA76-92-C-0009, July 1992.
 - [34] I. Weiss, "Geometric Invariants and Object Recognition", CAR-TR-632, N00014-91-J-1222 and DACA76-92-C-0009, August 1992.
 - [35] Y. Yacoob and L.S. Davis, "Early Vision Processing Using a Multi-Stage Diffusion Process", CAR-TR-633, DACA76-92-C-0009, August 1992.
 - [36] Z. Durić, A. Rosenfeld, and L.S. Davis, "Egomotion Analysis Based on the Frenet-Serret Motion Model", CAR-TR-634, AFOSR-91-0239, August 1992.
 - [37] P.J. Narayanan, "Effective Use of SIMD Machines for Image Analysis", CAR-TR-635, DACA76-92-C-0009, August 1992.

- [38] M. Abdel-Mottaleb, R. Chellappa, and A. Rosenfeld, "Passive Navigation Using Bayesian Estimation", CAR-TR-636, AFOSR-91-0239, August 1992.
- [39] A. Waksman and A. Rosenfeld, "Sparse, Opaque Three-Dimensional Texture, 2: Foliate Patterns", CAR-TR-637, DACA76-92-C-0009, September 1992.
- [40] E. Rignot, P. Dubois, and R. Chellappa, "Unsupervised Segmentation of Polarimetric SAR Data Using the Covariance Matrix", CAR-TR-638, F49620-92-J0130, September 1992.
- [41] T.-H. Wu and R. Chellappa, "Experiments on Estimating Motion and Structure Parameters Using Long Monocular Image Sequences", CAR-TR-640, DACA76-92-C-0009, October 1992.
- [42] Y. Yacoob and L.S. Davis, "Qualitative Labeling of Human Face Components from Range Data", CAR-TR-642, DACA76-92-C-0009, October 1992.
- [43] P.J. Rousseeuw, N.S. Netanyahu, and D.M. Mount, "New Statistical and Computational Results on the Repeated Median Line", CAR-TR-643, AFOSR-91-0239, October 1992.
- [44] E. Rivlin and I. Weiss, "Local Invariants for Recognition", CAR-TR-644, DACA76-92-C-0009 and F49620-92-J-0332, October 1992.
- [45] T.-H. Wu and R. Chellappa, "Stereoscopic Recovery of Egomotion and Environmental Structure: Models, Uniqueness and Experiments", CAR-TR-646, DACA76-92-C-0009, November 1992.
- [46] C. Fermüller and Y. Aloimonos, "The Role of Fixation in Visual Motion Analysis", CAR-TR-647, DACA76-92-C-0009 and IRI-90-57934, November 1992.
- [47] M. Abdel-Mottaleb, R. Chellappa, and A. Rosenfeld, "Binocular Motion Stereo using MAP Estimation", CAR-TR-650, F49620-93-1-0039, November 1992.
- [48] D.M. Mount and N.S. Netanyahu, "Computationally Efficient Algorithms for High-Dimensional Robust Estimators", CAR-TR-652, CCR-89-08901 and AFOSR-91-0239, December 1992.
- [49] A. Rosenfeld, "Image Analysis and Computer Vision: 1992", CAR-TR-653, F49620-93-1-0039, January 1993.
- [50] Y. Yacoob, L.S. Davis, and H. Samet, "Scale Space Interpolation and Metamorphosis", CAR-TR-654, DACA76-92-C-0009 and IRI-90-17393, January 1993.
- [51] S. Thompson and A. Rosenfeld, "Discrete Stochastic Growth Models for Two-Dimensional Shapes", CAR-TR-656, F49620-93-1-0039, January 1993.
- [52] N.S. Friedland, "Utilizing Energy Function and Description Length Minimization for Integrated Delineation, Representation, and Classification of Objects", CAR-TR-657, F49620-93-1-0039, January 1993.

Image Understanding Research at SRI International

Martin A. Fischler and Robert C. Bolles

Artificial Intelligence Center
SRI International

333 Ravenswood Ave., Menlo Park, CA 94025
(fischler@ai.sri.com bolles@ai.sri.com)

Abstract

The image understanding program at SRI International is a broad effort spanning the entire range of machine vision research. In this report we describe our progress in two major scientific areas: (1) recovery of scene geometry, and (2) semantic labeling and scene modeling. We have addressed the problems of automated and interactive scene analysis in two application domains: the first is concerned with modeling the earth's surface from aerial imaging sensors; the second is concerned with ground-level vision and vision-based land navigation.

In particular, we describe progress in automated and interactive scene modeling and visualization; in automatic image segmentation and delineation of both natural and man-made objects; in detecting and tracking moving objects; and in using knowledge beyond shape and immediate appearance to recognize objects in natural scenes and other complex domains.

1 Introduction

The overall goal of Image Understanding research at SRI International is to obtain solutions to fundamental problems in computer vision that are essential in allowing machines to model, manipulate, and understand their environment from sensor-acquired data and stored knowledge.

In this report we describe progress on the fundamental problems of geometric recovery and semantic labeling in two application domains: aerial and ground-based vision.¹ The first application domain is concerned with modeling the earth's surface from photographs taken from aircraft and satellites; the second application domain is concerned with modeling a natural environment in real time from data acquired by a robotic device moving through, and interacting with, this environment.

In both application domains, the underlying problems in *geometric modeling* involve making our recovery

techniques robust enough to operate in complex scene domains where conventional correlation based stereo is known to be inadequate (e.g., in urban scenes where occlusions prevent direct matching, and in ground-level views of natural terrain where there are radical depth changes over very small angular displacements and thus the continuity assumption is often violated). We describe progress in using multiple images to obtain an integrated geometric and physical description of the scene in terms of surfaces and their reflectance properties rather than just isolated points. We are also involved in an effort to assess the capabilities of existing stereo approaches and techniques to support land-based navigation, in developing techniques for building object descriptions that evolve gradually over time as more data are obtained, and in devising motion analysis techniques for detecting and tracking moving objects in data taken by moving sensors.

The problems of *semantic labeling* (e.g., object recognition) are known to be extremely difficult, and we have taken two different approaches in our work. Where it is feasible for a human to be part of the process, as is typically true in such tasks as building terrain and "site" models from aerial imagery, the problem becomes one of designing an effective interactive environment with the proper man-machine interface, feature extraction procedures, and visualization capabilities. In the robotic domain, where human intervention is limited, we have devised a context-based methodology for recognizing complex man-made and natural objects, and have developed and refined a set of procedures for delineating specified natural and man-made objects that are "line-like" in appearance (e.g., trees and roads).

An important theme in much of our current work is an emphasis on robustness and computational performance — especially through the development of algorithms capable of exploiting the parallel machine architectures now available (e.g., the Connection Machinetm).²

¹Supported by various Defense Advanced Research Projects Agency contracts.

²Use of a Connection Machine was provided by DARPA.

2 Geometric Recovery

The goal of geometric recovery is to build a three-dimensional structural description of a scene to support such tasks as robot navigation and cartographic modeling. Ideally the description would consist of several interconnected representations, including a detailed representation of the support surface (i.e., the ground), a list of material types and semantic labels for all scene "objects," and a set of accurate transformations from the local vehicle coordinate system to the global reference system. For many tasks, especially robot navigation, the process of building a scene model should be viewed as an on-going process in which a continuous stream of data is used for incrementally updating the representations. In practice, however, current scene modeling techniques typically analyze each snapshot of a scene independently and produce a loose patchwork of representations, including such things as ground surface patches, clouds of x-y-z points associated with objects, and a set of imprecise transformations from the local coordinate system to global system.

Our research goals in this area are to develop compact and expressive representations for modeling natural objects, such as rocks and bushes, and to develop effective techniques for incrementally compiling a complete scene model from multiple views. In this section we briefly describe our recent progress on the following topics that are steps toward our broader goals:

- Representation of objects as three-dimensional surfaces, instead of viewpoint-dependent two-and-a-half-dimensional surfaces.
- Extraction of both depth and reflectance maps by integrating stereo and photometric analysis.
- Use of temporal analysis to detect, track, and identify independently moving objects in a scene.
- Evaluation of current stereo techniques, including characterization of their strengths and weaknesses.

2.1 Three-Dimensional Object Models

In the past we have developed a number of three-dimensional object representations, including fractal-based descriptions [Pentland], contextual representations [Strat & Fischler], and a "representation space" approach [Bobick & Bolles]. Recently we have developed a triangulated mesh model that supports both object segmentation and surface refinement techniques. In the previous Image Understanding Workshop Proceedings we described a technique for coalescing clouds of three-dimensional points into a small number of representative surfaces [Fua and Sander]. In the past year we have concentrated on a specialization of the triangulation representation that we call hexagonally-connected

triangular meshes. These meshes have the advantage that they can be easily deformed to refine their local shape so that they satisfy both photometric and depth constraints. The use of these meshes as part of a technique for integrating stereo processing and photometric analysis is presented in a separate paper in these proceedings [Fua & Leclerc].

One advantage of a triangular mesh representation is that many computers now incorporate special hardware to support and perform graphics operations on such representations. This same hardware can be used, with appropriate analysis routines, to predict such things as scene depth values, surface orientations, and observed intensities. We have implemented some of our techniques on Silicon Graphics computers that support these operations.

Since these mesh representations are three-dimensional, they can directly encode all aspects of an object's appearance in a single structure. This structure, in conjunction with rendering techniques, provides a convenient way to work with complex, convoluted objects.

In most of our experiments, we have used regular meshes. While this is appropriate for surfaces whose properties remain relatively constant, it is not optimal for complex surfaces that require the combined efficiency and accuracy provided by irregular networks. The relatively smooth parts of such surfaces can be represented by large patches, while the rougher parts could be described by finer, more precise triangulations. We are in the process of implementing irregular networks formed by allowing selected facets to be subdivided.

2.2 Integration of Stereo and Photometric Analysis

Over the past few years we have investigated techniques for integrating stereo and photometric analysis because these two techniques are complementary; one works well when the imagery contains distinctive photometric patterns and the other works well when the imagery only contains gradual shading. In 1991 we reported the results of our first technique of this type [Leclerc & Bobick]. Recently we have developed a new approach that functions well even though we have relaxed several assumptions commonly used in shape-from-shading techniques. This new technique computes both the shape and reflectance properties of physical surfaces from the information present in multiple images. It considers two classes of information. The first is the information that can be extracted from a single image, such as texture gradients, shading, and occlusion edges. The technique takes advantage of the fact that multiple images enhance the utility of this type of information by allowing both consistency checks to filter out mistakes and averaging to improve precision. The second class of information includes the stereo depth values computed from two or

more images.

Our surface reconstruction method uses an object-centered representation, specifically, a hexagonally-connected three-dimensional mesh of vertices with triangular facets. Such a representation accommodates the two classes of information mentioned above, as well as multiple images (including motion sequences of a rigid object) and self-occlusions. We have chosen to model the surface material using a Lambertian reflectance model with variable albedo, though generalizations to specular surfaces are possible. Consequently, the natural choice for the monocular information source is shading, while intensity is the natural choice for the image feature used in multi-image correspondence. Not only are these the natural choices when we are able to assume a Lambertian reflectance model, they are complementary: intensity correlation is most accurate wherever the input images are highly textured, and shading is most accurate when the input images have smooth intensity variation. Since we wish to deal with surfaces with non-uniform albedo, we have developed a new approach that analyzes the facet-to-facet geometry and albedo pattern to recover surface models.

We use an optimization approach to reconstruct the surface shape and its albedo from the input images. We alter the shape and reflectance properties of the surface mesh to minimize an objective function, given an initial surface estimate provided by other means, such as a standard stereo algorithm. The objective function is a linear combination of an intensity correlation component, an albedo variation component, and a surface smoothness component. The first two components are a function of the intensities projected onto the triangular facets from the input images (taking occlusions into account), and are weighted according to the amount of texture in the intensities, for the reasons mentioned in the previous paragraph. The geometric smoothness component is slowly decreased during the optimization process to allow for an accurate final estimate of the surface shape and reflectance.

We have implemented an algorithm employing these three terms and have performed extensive experiments using synthetic images as well as aerial and face images. The strengths of the approach include:

- The use of the 3-D surface mesh allows us to deal with self-occlusions and thus effectively merge information from several potentially very different viewpoints to eliminate "blind-spots."
- By combining stereo and shape from shading, and weighing appropriately the reliability of their respective contributions, we can obtain results that are better than those produced by either technique alone.
- Using the facets to perform the stereo computation frees us from the constant-depth assumption that

standard correlation-based stereo techniques make. It becomes possible to recover accurately the depth of sharply sloping surfaces (such as that of a sharp ridge).

- The shape-from-shading component does not make the constant-albedo assumption common to most shading algorithms. Instead, we only make the weaker and much more general assumption that albedoes vary slowly across textureless areas.

More complete details of this technique can be found in a separate paper in these proceedings [Fua & Leclerc].

2.3 Moving Object Detection, Tracking, and Recognition

Our goal in this research effort is to develop automated methods for producing three-dimensional models of scenes containing moving objects. Our approach is to analyze sequences of temporally coherent images, because they provide the machine with both "redundant" information and new information about the scene. The redundant information can be used to increase the precision and reliability of computed models; the new information can be used to extend models into previously unseen areas. Recently we have developed two new techniques of this type. One is a real-time technique designed to provide feedback within an "active vision" paradigm. The other integrates object recognition into the tracking process in order to bridge gaps in tracking-continuity caused by such things as occlusion and low-level processing mistakes.

The first technique, which is the product of a joint effort between Xerox PARC, Stanford University, and SRI, produces motion results (or stereo disparities) at 10 to 15 hertz. It has been implemented on two multi-processor configurations, a 16k-processor Connection Machine and a 5-processor VX/MVX graphics accelerator system (200-MIPS). With these systems we have demonstrated real-time control of a five degree-of-freedom camera system tracking a person walking around a room [Woodfill].

The second technique incorporates object recognition procedures into the tracking process in order to improve tracking reliability and facilitate object identification. Our strategy has involved four steps. First, we "train" the system to recognize an object, such as a truck, by showing it to the system from several viewpoints. Second, given an image sequence of the truck moving in front of the camera system, we apply our "weaving-wall" tracking technique [Baker & Garvey] to build a temporal model of the moving objects in the sequence. Third, we apply the PRS recognition system [Chen & Mulgaonkar] to identify the truck in individual images. And fourth, we use the recognition results to "explain" discontinuities in tracking the various objects so that we can pro-

duce a more coherent description of the motion in the scene.

2.4 Evaluation of Current Stereo Techniques

Stereo analysis, which for a long time had been viewed as an interesting, but too-costly-to-be-practical technique, has emerged as a viable tool for realtime applications, such as vehicle navigation. This has happened for two reasons. First, advances in hardware have made it practical to compute stereo matches "in real time." And second, advances in algorithm development have made it possible to correctly match large portions of outdoor scenes.

An important next step in the development and use of practical stereo systems is the characterization of their capabilities. Potential users, such as system integrators and automatic task-planners, need to know the techniques' computational requirements, their speeds, the precision of their results, their common mistakes, etc. in order to model the behavior of these stereo systems and reason about their use. With this in mind, SRI, JPL, and Teleos began a multi-phase evaluation process last year within the DARPA Unmanned Ground Vehicle (UGV) Project. The first phase of that evaluation has been completed and the second phase has begun.

The overall plan for that evaluation was (and continues to be) to pursue a three-pronged approach, including analytic models, qualitative "behavioral" models, and statistical performance models. The analytic models would be used to estimate such things as the expected depth precision computable with a specific camera configuration. The qualitative models would be used to identify key problems for future research, for example, detection of holes, analysis of shadowed regions, and depth measurements in bland areas. The statistical models would be used to produce quantitative estimates of such key factors as the smallest obstacle detectable at a specified distance. SRI has taken the lead in the qualitative evaluation; JPL has taken the lead in the quantitative analysis.

For the qualitative analysis we decided to start by examining a small number of techniques in order to debug the process, and then expand the evaluation to include a much larger set of participants. The goals of the first phase were to get an initial estimate of the effectiveness of current stereo techniques applied to UGV tasks, to identify key problems for future research, and to debug the evaluation process.

One of the high-level guidelines we adopted was to develop and maintain an atmosphere of cooperation and constructive criticism among the researchers participating in the evaluation. Without this we would not be able to focus on our ultimate goal of producing a sequence of increasingly capable stereo systems. To help

establish a cooperative atmosphere we decided to concentrate on the positive aspects of each technique and emphasize potential extensions, realizing that existing techniques were developed for different domains and different applications. We also decided to share all the raw results with the participants so they could duplicate our analysis or develop their own.

For the first phase of the qualitative evaluation, SRI collected imagery from five groups, JPL, INRIA (in France), SRI, CMU, and Teleos (hence the name "JISCT" for the first evaluation phase); selected 49 image pairs for analysis; converted them into a standard format; distributed the dataset to the five groups for processing, along with an extensive set of instructions; collected the results; characterized them; and finally distributed the results and the associated report to the participants.

We intentionally asked each group to process a large number of pairs (10 training pairs and 45 "test" pairs ... 6 pairs were in both the training and test sets), because we wanted to force each group to establish a standard algorithm that was automatically applied. As result of this approach, there are now 3 or 4 groups around the world that can readily apply end-to-end stereo techniques to new data and compare their results. As part of the second phase we hope to expand this community to 10 or more groups. This process has opened up a new form of interaction within the computer vision community that we feel will help stimulate advances and reduce redundant development.

In the instructions to the participants we asked each group to produce several results for each match point in addition to its computed disparity. For each point we asked for an x and a y disparity, an estimate of the precision associated with each reported disparity, an estimate of the confidence associated with each match, and an annotation for each unmatched point, indicating why the technique could not find a match. Possible explanations for no match included "area too bland," "multiple choices," and "inconsistent with neighbors." Although none of the groups produced all this additional information (they all produced some of it), we felt that it was important to begin the process with the goal of producing this auxiliary information, which will be invaluable for the higher-level routines using the stereo results. We foresee a time in the not too distant future when the calling routine will use the precisions, confidences, and annotations to actively control the sensor parameters for the next data acquisition step. For example, if the current stereo results contain a large region of no disparities and the image regions are quite dark, the controlling routine could open the irises or increase the integration time to re-examine these dark regions.

To assist in the analysis of the results, SRI developed two sets of routines, one to gather statistics and one to display the disparities in a variety of ways. Since we

did not have ground truth for the distributed imagery, we were not able to compare the computed disparities with objective values. However, we were able to gather statistics on two of the three types of mistakes that we were interested in by outlining selected regions in the imagery and counting the occurrence of results/no-results within these regions. We made a distinction between the following three types of mistakes:

False Negatives: No disparities computed for points that should have results.

False Positives in Unmatchable Regions: Disparities reported for points that don't have matches in the second image, for example, points occluded in one image or points out of the field of view of one of the images.

False Positives in Matchable Regions: Incorrect disparities reported for matchable points.

By interactively outlining regions of occluded points, regions of points out of the field of view of the second image, and regions of points in the sky, we were able to directly measure statistics for the first two types of mistakes. In addition, we outlined regions corresponding to expected problems, such as dark shadows, foliage, and bland areas. In this way we could gather statistics on the behavior of the algorithms on these special problems.

A high-level summary of the results of the first phase evaluation are as follows:

- We were surprised by the completeness of the results. Even though the dataset contained a wide range of imagery, including some sequences designed to stretch the analysis along specific dimensions, such as noise tolerance and disparity range, the stereo systems computed disparities for 64% of the matchable points. On 8 image pairs selected to be the most appropriate for UGV applications, the techniques computed disparities for up to 87% of the points. Although the missing points (and mistakes in the reported matches) could cause problems for vehicle navigation, this level of completeness is an indication that there is a solid basis for building a passive ranging system for an outdoor vehicle.
- For the UGV-related imagery the number of gross errors was relatively small, ranging from a few "spike" errors to small regions of mistakes. We estimate that there were between 1 and 5% gross errors in these results. Many of these errors would have to be eliminated in order for the data to be used directly for planning navigable routes.
- The stereo systems made different mistakes, most of which could be explained by their correlation patch size, search technique, or match verification technique. However, since they made different mistakes,

there is a possibility of combining them in a way to check each other and fill in missing data.

- All the stereo systems could be improved significantly with a relatively small amount of effort. This was the first test of this type, requiring the analysis of a large dataset, and it uncovered some weaknesses in the different stereo systems that can be corrected. One area to be considered is the development of pre-analysis techniques to automatically set key parameters, such as patch size and search areas (as Teleos did). Another place for improvement is in the filtering of results, eliminating matches that differ significantly from their neighbors (as SRI did).
- There were a few surprises, such as Teleos's successful solution to one set of image pairs from CMU that includes a carpet with a repetitive pattern on it. Teleos's large patches were able to detect large regions of subtle differences, which allowed recovery of the correct disparities.

Additional information about the JISCT evaluation, its results, and our goals for the second phase, can be found in a separate paper in these proceedings [Bolles, Baker, & Hannah].

3 Interactive Modeling

Our work in the area of Interactive Modeling is concerned with the development of an interactive environment and associated feature extraction and visualization techniques to enable effective human assisted scene/site model construction - especially for applications in the areas of cartography, intelligence analysis, and mission planning.

3.1 Interactive Techniques for Scene Modeling: A Cartographic Modeling Environment

Manual photointerpretation is a difficult and time-consuming step in the compilation of cartographic information. However, fully automated techniques for this purpose are currently incapable of matching the human's ability to employ background knowledge, common sense, and reasoning in the image-interpretation task. Near-term solutions to computer-based cartography must include both interactive extraction techniques and new ways of using computer technology to provide the end-user with useful information in the form of both image and map-like interactive computer displays.

In order to support research in semiautomated and automated computer-based cartography, we have developed the SRI Cartographic Modeling Environment (CME). In the context of an interactive workstation-based system, the user can manipulate multiple images;

camera models; digital terrain elevation data; point, line, and area cartographic features; and a wide assortment of three-dimensional objects. Interactive capabilities include free-hand feature entry, feature editing in the context of task-based constraints, and adjustment of the scene viewpoint. Synthetic views of a scene from arbitrary viewpoints may be constructed using terrain and feature models in combination with texture maps acquired from aerial imagery. This ability to provide an end-user with an interactively controlled scene-viewing capability could eliminate the need to produce hard-copy maps in many application contexts. Additional applications include high-resolution cartographic compilation, direct utilization of cartographic products in digital form, and generation of mission-planning and training scenarios.

In cooperation with General Electric, SRI has enhanced the CME to fully support the needs of the RADIUS Program (as described below).

3.1.1 The RADIUS Common Development Environment

Progress in image understanding has often been hampered by the difficulty involved in sharing results and software among collaborating institutions. This problem is being addressed within the RADIUS Program through development of a common development environment. RADIUS is a government sponsored program whose aim is to support image analysts through the construction and use of site models. Image understanding techniques are expected to play a major role, in both the extraction of cartographic features to populate site models, as well as in the employment of site models in more detailed scene analysis. Transfer of research results within RADIUS is being facilitated through the use of a common development environment, based on SRI's Cartographic Modeling Environment (as described above).

The resulting environment has been named the RADIUS Common Development Environment (RCDE). The RCDE has been distributed to participants in the RADIUS Program and is now available for distribution to all members of the DARPA Image Understanding community. Widespread adoption of the RCDE throughout the IU community has the potential for pay-offs at 3 levels:

- Sharing research results among the various research laboratories to foster collaborative work and to build on the successes of others.
- Validation of research results by other laboratories to insure the soundness of the work, to compare alternative techniques, and to motivate further investigations.
- Technology transfer from research laboratories to development organizations which also utilize the common development environment.

The RCDE is the culmination of many years of re-

search at SRI and GE on interactive programming environments for image understanding. SRI and GE are pleased to be able to make it available for use by other image understanding laboratories. (See references [CME91, RCDE92, RCDE93a, and RCDE93b].)

3.2 Terrain Modeling and Visualization

The DARPA sponsored MAGIC (Multidimensional Applications and Gigabit Internetwork Consortium) project has been established to develop a very high-speed, wide-area networking testbed that will demonstrate real-time, interactive exchange of data at gigabit-per-second (Gbps) rates among multiple distributed servers and clients. Participants in the project include a variety of organizations from government, industry, and academia.

The SRI role in this project includes the design and implementation of the MAGIC terrain visualization application, as well as the production of the digital elevation models (DEMs) and high-resolution ortho-rectified aerial images (ortho-images) used by the application.

This application will allow a U.S. Army commander to "fly over" or "walk/drive through" a battlefield at his or her own speed in real time. Views of the (on-going) battle available for selection will range from low-resolution, wide-area coverage to high-resolution, limited-area coverage and will include fixed and mobile objects such as buildings and vehicles. The positions of mobile objects will be updated in real time.

What makes this application unique is that the area of interest is quite large (tens to hundreds of gigabytes), and that the data must be accessed across a high-speed network. In the first phase of the project, to be completed by the end of 1993, the area of interest is a 900 sq. km. exercise area of the National Training Center at Fort Irwin, California. The full-color ortho-images alone, at one meter ground-level resolution, require 2.7 gigabytes of storage. The next phases will involve much larger areas.

The size of the database, the need for real-time network access, and the wide range of views has led us to represent the ortho-images and DEMs as a "tiled Gaussian pyramid." That is, the ground-level distance between pixels doubles from one level to the next (starting with one meter resolution for the ortho-images, and 32 meters for the DEMs), and each level is divided into equal-size image windows called "tiles." Using equal-size tiles (initially planned to be 128 x 128 pixels) facilitates the storage, transmission, and display of the data.

Although the network itself has a very high bandwidth, there are inherent delays in requesting tiles from a disk-based storage system (ISS) across large distances. Consequently, the application must request tiles well in advance of their being required for display. This requires processes for predicting future viewpoints, determining which tiles are visible from a particular viewpoint, re-

questing these visible tiles in an appropriately prioritized fashion, and receiving and buffering the tiles in a shared-memory data structure accessible to the display process.

Also, the application must be able to deal with lost and/or delayed tiles in a relatively seamless fashion. This is accomplished in two ways. First, the tiled Gaussian pyramid representation allows the entire area of interest to be represented at some coarse resolution using a relatively small number of tiles. These coarse-resolution tiles can reside entirely in the rendering engine, thereby guaranteeing that any viewpoint can be rendered instantly, no matter how radical the change in viewpoint, albeit at a coarser resolution than perhaps desired.

Of course, it is not sufficient to merely guarantee that any viewpoint can be displayed at any time at some coarse resolution. The tile pre-fetch process must attempt to have all of the appropriate resolution tiles in memory when they are needed for display. This is done by predicting the user's path, and searching for tiles in all of the views along that path from the current moment to some future moment. (In fact, as one attempts to predict further into the future, the "view" needs to become more encompassing in order to account for the increasing uncertainty in the predicted path.) Tiles from any of these views that are not currently in memory are added to the list of tiles to be requested from the ISS using a coarse-to-fine search algorithm, where coarser-resolution tiles are given a higher priority than finer-resolution tiles.

Searching all views in the above manner, and requesting all visible tiles that are not currently in memory in a coarse-to-fine fashion has several consequences. First, in the worst case of a change to a completely new viewpoint, one will immediately be able to display a coarse resolution representation of the scene followed by increasingly finer resolution representations. (It is currently expected that this should occur over at most a few frames.) Second, the coarse-to-fine ordering of the requests should prove useful even when the user is not radically changing viewpoints because it will increase the likelihood of intermediate resolution tiles being available, since these cover a larger area than the finest resolution tiles, and will be requested with a higher priority. Third, since tiles not currently in memory are always requested, lost or delayed tiles are automatically re-requested without the need for special protocols between the ISS and the application.

We see our work on the MAGIC project, not only as a specific application of technology developed in the IU program, but also as providing important new tools needed for the visualization component of our efforts in interactive scene modeling and IU related applications.

3.3 Additional Topics In Interactive Scene Analysis

We recently made a significant advance in the long-standing problem of duplicating human performance in recovering 3-D models of objects with straight edges and planar faces from qualitative and imprecise line drawings (e.g., building edges as in a single approximate projection of the corresponding wire-frame). This work can greatly simplify communication problems between man and machine in such applications as robotic mission planning and in construction of databases for use in robotic navigation. A paper describing this work has been published in the International Journal of Computer Vision [Leclerc & Fischler]. Our goal in this on-going work is to extend the basic approach to both curved-line drawings (e.g., of terrain elevations as in an approximate and uncalibrated contour map) and to sketches and actual images of natural terrain. The key elements of our current optimization-based approach are a way to automatically extract planar constraints from the line drawings; an objective function that honors but does not insist on the planarity constraints in combination with terms measuring regularity and symmetry of proposed solutions; and a continuation technique (we developed) to find the solution. For the limited class of 20-30 objects we used in our experiments, we have been able to consistently obtain the desired solution for the same object in almost all of its possible projections.

4 Semantic Labeling, Partitioning, and Delineation

The natural outdoor environment poses significant obstacles to the design and successful integration of the interpretation, planning, navigational, and control functions of a robotic device supported by a general-purpose vision system. Many of these functions cannot yet be performed at a level of competence and reliability necessary to satisfy the needs of an autonomous robot. The problem lies in the inability of available techniques, especially those involved in sensory interpretation, to use contextual information and stored knowledge in recognizing objects and environmental features, inability to provide adequate models to the machine as a basis for recognizing complex man-made and natural objects, and the lack of an adequate collection of low-level feature extraction techniques capable of robust performance over a wide range of scene domains.

Our current work in this topic area, presented below, exploits three key ideas:

- (1) use of an objective function and optimization as a descriptive mechanism,
- (2) use of evolving context in a production system formalism to select feature extraction techniques and pa-

parameter settings in a knowledge-based paradigm for low-level vision and interactive modeling, and

(3) use of a few highly refined and reliable low-level techniques as the base for a much wider class of feature extraction methods.

In this section we discuss the development of techniques for automatically recognizing and delineating complex man-made and natural objects, especially for applications to robotic navigation in the outdoor world, and to provide robust automated techniques for aerial image analysis.

4.1 Condor: A Context Based Approach to Scene Modeling

Much of the progress that has been made to date in machine vision has been based, almost exclusively, on shape comparison and classification employing locally measurable attributes of the imaged objects (e.g., color and texture). Natural objects viewed under realistic conditions do not have uniform shapes that can be matched against stored prototypes, and their local surface properties are too variable to be unique determiners of identity. The standard machine vision recognition paradigms fail to provide a means for reliably recognizing *any* of the object classes common to the natural outdoor world (e.g., trees, bushes, rocks, and rivers). In this effort [Strat&Fischler], we have devised a new paradigm that explicitly invokes context and stored knowledge to control the complexity of the decision-making processes involved in correctly identifying natural objects and describing natural scenes.

The conceptual architecture of the system we describe, called Condor (for context-driven object recognition), is much like that of a production system; there are many computational processes interacting through a shared data structure. Interpretation of an image involves the following four process types.

- Candidate generation (hypothesis generation)
- Candidate comparison (hypothesis evaluation)
- Clique formation (grouping mutually consistent hypotheses)
- Clique selection (selection of a "best" description)

Each process acts like a daemon, watching over the knowledge base and invoking itself when its contextual requirements are satisfied. The input to the system is an image or set of images that may include intensity, range, color, or other data modalities. The primary output of the system is a labeled 3D model of the scene. The labels included in the output description denote object *classes* that the system has been tasked to recognize, plus others from the recognition vocabulary that happen to be found useful during the recognition process. An

object *class* is a category of scene features such as sky, ground, geometric-horizon, etc.

Visual interpretation knowledge is encoded in *context sets*, which serve as the uniform knowledge representation scheme used throughout the system. The invocation of all processing operations in Condor is governed by context through the use of various types of context sets: an action is initiated only when one or more of its controlling context sets is satisfied. Thus, the actual sequence of computations, and the labeling decisions that are made, are dictated by contextual information, by the computational state of the system, and by the image data available for interpretation.

The successful processing of a significant number of outdoor images has demonstrated the validity and importance of the Condor paradigm. Our continuing work on Condor addresses the problems of (1) how to efficiently construct (or acquire) the large site-specific database needed for successful operation, and (2) how to improve the effectiveness of a few key low-level routines that Condor depends on. We have also modified and extended the Condor paradigm to permit its use in an interactive environment; this work is presented in a separate paper in these proceedings [Strat] and briefly discussed in the following paragraphs.

The semiautomated nature of RADIUS (see earlier discussion of the RADIUS program) obviates the need for some of the machinery employed in the fully automated version of Condor. The availability of a human operator permits access to some kinds of context that were not available to Condor, such as the level of interactivity desired and manual sketches of individual features. The existence of a human to review and edit the IU results offers the opportunity to use a supervised learning scheme to improve the quality of the knowledge base and to extend its range of competence.

The large number of features and wide range of imaging conditions that must be considered for site-model construction in RADIUS stress the context set representation employed in Condor. While context sets were adequate to represent Condor's knowledge base, it has been necessary to consider more effective representations that will extend to the requirements of site-model construction. Two new constructs, context tables, and context rules, offer a more systematized organization for the context knowledge base that should facilitate its construction. These representations offer additional economies in both storage and computation that may be vital to implementation of large systems. The symmetry of context tables and rules encourage their use in either direction: to select algorithms and set their parameters, as well as to describe the conditions that must be satisfied for a given algorithm to be applicable. This final capability raises the possibility of using context rules to choose the most appropriate images for interpretation.

4.2 Optimization-Based Methods for Partitioning, Delineation and Recognition

It is commonly accepted that the problems of partitioning, delineation, and recognition require non-local information for their solution. Optimization is one of the major paradigms for combining and evaluating global information, but to use this paradigm one must address the issues of how to select an objective function that accurately reflects the intended solution, and how to devise a procedure for the optimization process that will return an answer with a practical amount of computation.

In previous reports we described two lines of research using optimization-based methods for partitioning and delineation:

- 1) An optimization-based approach, applicable both to image partitioning and to subsequent steps in the scene analysis process, that involves finding the "best" description of the image in terms of some specified descriptive language. In the case of image partitioning [Leclerc88,89a,89b,89c,89d,90a,90b,91], we employ a language that describes the image in terms of regions having a low-order polynomial intensity variation plus white noise; region boundaries are described by a differential chain code. The best description is defined as the simplest one (in the sense of least encoding length) that is also stable (i.e., minor perturbations in the viewing conditions should not alter the description). A continuation method, specially designed to match the problem constraints, was used to solve the optimization problem.

- 2) In situations where the required image description must extend beyond that of a delineation of coherent regions, we require an extended vocabulary relevant to the semantics of the given task. Fua and Leclerc deal with the problem of boundary/shape detection given a rough estimate of where the boundary is located and a set of photometric (intensity-gradient) and geometric (shape-constraint) models for a given class of objects [Fua&Leclerc88, Fua&Leclerc90]. They define an energy (objective) function that assumes a minimal value when the models are exactly satisfied. An initial estimate of the shape and location of the curve is used as the starting point for finding a local minimum of the energy function by embedding this curve in a viscous medium and solving the dynamic equations. This energy-minimization technique, has been applied to straight-line boundary models and to more complex models that include constraints on smoothness, parallelism, and rectilinearity. In an interactive mode, the user supplies an initial estimate of the boundary of some object (which may be quite complex, like the outline of an aeroplane) and then, if need be, corrects the optimized curve by applying forces to the curve or by changing one of a few optimization/model parameters.

We believe that the above techniques represent signif-

icant advances in the state-of-the-art in their respective areas of image partitioning and delineation. The implemented systems based on these techniques have been able to produce excellent results in complex situations where existing (typically local) approaches often fail. In the following subsection we describe on-going work which employs the above techniques under a Condor like control structure to deal with the problem of efficient site-model construction.

4.2.1 Model-Based Optimization for Site Modeling

As part of our work on the DARPA sponsored RADIUS Program (described earlier) our research seeks to increase the speed and accuracy with which site models can be constructed from available imagery by developing a new family of image understanding (IU) techniques.

Model-Based Optimization (MBO) is a paradigm in which an objective function is used to express both geometric and photometric constraints on features of interest. A model of a feature (such as a road, a building, or coastline) is extracted from an image by adjusting the model until a minimum value of the objective function is obtained. The optimization procedure yields a description that simultaneously satisfies (or nearly satisfies) all geometric and image constraints, and, as a result, is likely to be a good model of the feature.

The applicability of MBO is currently limited by the expressive power of terms in the objective function and by the difficulty of optimization. We are attempting to extend the range of objects that can be modeled within the MBO paradigm, and to develop suitable optimization procedures to support their extraction from imagery.

With these extensions, the MBO technology can be used (1) automatically, to extract from an image all objects of a given type, or (2) interactively, to extract objects that are of special interest or that were missed by a fully automated system.

This research seeks not only to develop new IU techniques for cartographic feature extraction, but also to develop a language with which an image analyst can communicate with such a system. The foundation for this language lies in the creation and implementation of a large number of feature extraction operations, each of which is sensitive to the context of a particular task.

To this end, we have developed a means to utilize contextual information (see discussion of Condor) to automatically produce IU operators that are tailored to the specific extraction problem of interest, and hence are more likely to succeed than generic IU algorithms. A description of this approach appears as a separate paper in this proceedings [Strat].

The technology is currently being implemented as a customized system built using the RADIUS Common

Development Environment (RCDE). Its design is being shaped through experiments using multiple overhead images in a site model construction scenario. The benefits of the new technology will be measured by comparing its performance on site-model construction tasks with that achievable using other manual and semiautomated techniques.

4.3 Curve Partitioning and Delineation of Man-Made and Natural Objects

We have identified a few key low-level routines, partitioning and delineation algorithms in particular, that if made sufficiently robust, could form the basis of a wide variety of feature extraction algorithms. In addition to our optimization-based methods, we have made new progress in extending some of our past work in this problem domain.

A critical problem in machine vision is how to break up (partition) the perceived world into coherent or meaningful parts prior to knowing the identity of these parts. Almost all current machine vision paradigms require some form of partitioning as an early simplification step to avoid having to resolve a combinatorially large number of alternatives in the subsequent analysis process.

Finding salient points on image curves (potential partition points) plays a critical role in both two and three dimensional object recognition, in curve approximation, in tracking moving objects, and in many other tasks in machine vision. For example, in cartography, computer graphics, and in many scene analysis tasks, it is often desirable to partition an extended boundary or a contour into a sequence of simply represented primitives (e.g., straight line segments or polynomial curves of some higher degree) to simplify subsequent analysis and to minimize storage requirements. "Corners" on the contours of imaged objects are often used as features for tracking the motion of these objects and for computing optical flow.

In a paper (by Fischler and Wolf, appearing elsewhere in these proceedings) describing our current work in this topic area, we present the underlying ideas and algorithmic details of a computer program (the SSS algorithm) that performs at a human level of competence for a significant subset of the *curve partitioning* task. It extends and "rounds out" the technique and philosophical approach originally presented in a 1986 paper by Fischler and Bolles. In particular, it provides a unified strategy for selecting and dealing with interactions between salient points, even when these points are salient at "different scales of resolution." Experimental results are described involving on the order of 1000 real and synthetically generated images.

A technique [Fischler & Wolf] was developed for detecting and delineating low resolution linear structures appearing in aerial imagery, such as roads and rivers.

The algorithm was effective in finding such structure, but it provided no mechanism for distinguishing between the semantically meaningful objects and the "accidental" and irrelevant linear features found in most real images. In related work now in progress to automatically detect and delineate roads in aerial images, we use the SSS algorithm to "slice up" the individual curves found by our existing delineation algorithm. We throw away the very small resulting segments which are typical of accidental linear formations, and then further filter the longer segments with respect to a set of semantic constraints. Those segments that pass through the filtering process are then "glued" back together to produce the desired delineation. The robustness of the SSS algorithm is essential in carrying out the filtering operation. Insertion of extraneous partition points would cause the loss of portions of the road network; absence of partition points would allow meaningless appendages to become part of the extracted network.

5 Acknowledgment

The following researchers have contributed to the work described in this report: H.H. Baker, R.C. Bolles, M.A. Fischler, P. Fua, M.J. Hannah, A. Heller, S. Lau, Y. Leclerc, L.H. Quam, T.M. Strat, and H.C. Wolf.

6 References

The following recent publications result fully or in part from SRI's image-understanding research program.

1. Baker, H.H. and T.D. Garvey, "Motion Tracking on the Spatiotemporal Surface," *Proc. IEEE Motion Vision Workshop*, Princeton, New Jersey, October 1991, (also in *Proc. of DARPA Image Understanding Workshop*, San Diego, California, January 1992.)
2. Barnard, S.T. and M.A. Fischler, "Computational and Biological Models of Stereo Vision," to appear in *Wiley Encyclopedia of AI*, 2nd ed; (also in *Proc. of DARPA Image Understanding Workshop*, Pittsburgh, Pennsylvania, pp. 449-455, September 1990).
3. Bobick, A.F. and R.C. Bolles, "The Representation Space Paradigm of Concurrent Evolving Object Descriptions," *IEEE PAMI*, Vol. 14, No. 2, pp. 146-156, February 1992.
4. Bolles, R.C., H.H. Baker, and M.J. Hannah, "The "JISCT" Stereo Evaluation," in *these proceedings*.
5. Chen, C-H. and P. G. Mulgaonkar, "Uncertainty Update and Dynamic Search Window for

- Model-Based Object Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Maui, Hawaii, 692-694, June 1991.
6. CME91, "Capabilities of the Cartographic Modeling Environment. General Electric and SRI International," July 1991.
 7. Fischler, M.A. and H.C. Wolf, "Linear Delineation," *Proc. IEEE CVPR-89*, pp. 351-356, June 1983.
 8. Fischler, M.A. and H.C. Wolf, "Locating Perceptually Salient Points on Planar Curves," accepted for publication in IEEE PAMI.
 9. Fischler, M.A. and H.C. Wolf, "Saliency Detection and Partitioning Planar Curves," included in *these proceedings*.
 10. Fischler, M.A. and Strat, T.M., "Recognizing Objects in a Natural Environment: A Contextual Vision System (CVS)," to appear in *Automatic Object Recognition*, edited by Hatem Nasr, 1991, (also in *Proc. of DARPA Image Understanding Workshop*, Palo Alto, California, pp. 774-796, May 1989).
 11. Fischler, M.A. and Y.G. Leclerc, "Recovering 3-D Wire Frames from Line Drawings," *Proc. of DARPA Image Understanding Workshop*, San Diego, California, January 1992.
 12. Fua, P.V., "Combining Stereo and Monocular Information to Compute Dense Depth Maps that Preserve Depth Discontinuities," in *Proc. of IJCAI 91*, Sydney, Australia, August 1991.
 13. Fua, P.V., "A Parallel Stereo Algorithm that Produces Dense Depth Maps and Preserves Image Features," to appear in *Machine Vision and Applications*, 1991.
 14. Fua, P.V. and P. Sander, "Reconstructing Surfaces from Unstructured 3D Points," *Proc. of DARPA Image Understanding Workshop*, San Diego, California, January 1992.
 15. Fua, P.V. and Y. Leclerc, "Object-Centered Surface Reconstruction: Combining Multi-Image Stereo and Shading," in *these proceedings*.
 16. Hannah, M.J., "A System for Digital Stereo Image Matching," *Photogrammetric Engineering and Remote Sensing*, Vol. 55, No. 12, December 1989, pp. 1765-1770.
 17. Leclerc, Y.G., "Constructing Simple Stable Descriptions for Image Partitioning," *Proc. DARPA Image Understanding Workshop*, Cambridge, Massachusetts, pp. 365-382, April, 1988.
 18. Leclerc, Y.G., "Constructing Simple Stable Descriptions for Image Partitioning," *International Journal of Computer Vision*, Vol. 3, No. 1, 1989a.
 19. Leclerc, Y.G., "Segmentation via Minimal-Length Encoding on the Connection Machine," *Fourth International Conference on Supercomputing*, Santa Clara, California, April 30-May 5, 1989b.
 20. Leclerc, Y.G., "Image and Boundary Segmentation via Minimal-Length Encoding on the Connection Machine," *Proc. DARPA Image Understanding Workshop*, Palo Alto, California, pp. 1056-1069, May, 1989c.
 21. Leclerc, Y.G., "The Local Structure of Image Intensity Discontinuities," Ph. D. Thesis, McGill University, Montréal, Québec, Canada, May, 1989d.
 22. Leclerc, Y.G., "Simplicity of Description as the Basis for Visual Interpretation," Working Notes of the AAAI Spring Symposium on The Theory and Application of Minimal-Length Encoding, Stanford University, Palo Alto, California, March 1990.
 23. Leclerc, Y.G., "Region Grouping using the Minimum-Description-Length Principle," *Proc. DARPA Image Understanding Workshop*, Pittsburgh, Pennsylvania, September 1990.
 24. Leclerc, Y.G. and A.F. Bobick, "The Direct Computation of Height from Shading," *Proc. 1991 Computer Society Conference on Computer Vision and Pattern Recognition*, Lahaina, Maui, Hawaii, June 1991, (also in *Proc. of DARPA Image Understanding Workshop*, San Diego, California, January 1992.)
 25. Leclerc, Y.G. and M.A. Fischler, "An Optimization Based Approach to the Interpretation of Single Line Drawings as 3-D Wire Frames," *IJCV* 9(2):113-136, November 1992.
 26. Lowrance, J.D., E.H. Ruspini, and T.M. Strat, "Understanding Evidential Reasoning," SRI Technical Note 501, January 1991, (also submitted to *International Journal of Approximate Reasoning*, December 1990).
 27. Mundy, J., Quam, L., Strat, T. Bremner, B., Horwedel, M., and Welty, R., "The RADIUS Common Development Environment," *Proc. AIPR*, Washington, DC, October 1991, (also in *Proc. of DARPA Image Understanding Workshop*, San Diego, California, January 1992.)
 28. Pentland, A.P., "Fractal-Based Description of Natural Scenes," *IEEE PAMI* Vol.6, No.6, pp. 661-674, 1984.

29. Quam, L. and T.M. Strat, "SRI Image Understanding Research in Cartographic Feature Extraction," *Proc. ISPRS2*, Munich, Germany, September, 1991 (also available as Tech Note 505, Artificial Intelligence Center, SRI International).
30. RCDE92, "RCDE User Operations Guide," General Electric and SRI International, April 1992.
31. RCDE93a, "RADIUS Common Development Environment User's Manual," General Electric and SRI International, January 1993 (Draft).
32. RCDE93b, "RADIUS Common Development Environment Programmer's Reference Manual," General Electric and SRI International, January 1993 (Draft).
33. Strat, T.M. "Decision Analysis Using Belief Functions," *International Journal of Approximate Reasoning*, Vol. 4, No. 5, pp. 391-418, September 1990.
34. Strat, T.M. and M.A. Fischler, "Natural Object Recognition: A Theoretical Framework and Its Implementation," *Proc. IJCAI-91*, Sydney, Australia, August 1991.
35. Strat, T.M. and M.A. Fischler, "Context-Based Vision: Recognizing Objects using both 2D and 3D Imagery," *IEEE PAMI* Vol. 13(10):1050-1065, October 1991.
36. Strat, T.M., "Employing Contextual Information in Computer Vision," *in these proceedings*.
37. Woodfill, J., "Motion Vision and Tracking for Robots in Unstructured Environments," Ph.D. thesis, Computer Science Department, Stanford University, August, 1992.

Image Understanding Research at CMU

Steve Shafer, Takeo Kanade, and Katsu Ikeuchi

School of Computer Science
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh PA 15213

Abstract

At CMU, research in Image Understanding spans the range of topics from the basic science of imaging to applications in autonomous intelligent systems. The focal areas are:

- *Physics-Based Vision*
- *3D Shape and Motion Recovery*
- *Computational Range Sensor*
- *Parallel Vision*
- *Vision for Object Recognition*
- *Vision for Robot Vehicles*
- *Vision for Human-Computer Interaction*

1. Physics-Based Vision

While many vision systems have been demonstrated in principle, few have been highly reliable when deployed. This is largely because of the reliance on feature detectors such as edge-finding, which are based on over-simple approximations to the principles of imaging physics. The detailed study of the imaging process can lead to vision systems with both greater power and more reliability. At CMU, we have pioneered in the exploration of physics-based vision, and our work in this area has led to major advances in deployed vision systems. We continue to study these fundamental issues, particularly in the measurement of object color, shape, and surface roughness.

The basic principle of physics-based vision is that most (non-metallic) objects display two colors: the "object color" of the material, and highlights caused by reflection from the outer surface. The object color is important for identifying objects, whereas highlights reveal the smoothness or roughness of the surface. Both can be used to identify the object shape and material. Metals display only highlights (i.e. surface reflection).

1.1. Color and Highlights

In 1984, CMU vision researchers showed that the

colors of pixels in an image of a non-metal ("inhomogeneous dielectric") have an important linear relationship to the two types of reflection. We made the prediction that the colors should lie on a plane in the RGB color space, and could be analyzed by linear algebra to actually measure the amounts of each reflection component at each pixel. In 1987, we accomplished this analysis, and found that the pixel colors did not fill the plane in RGB space, but formed a "skewed-T" shape.

We (Novak and Shafer) have now added to this theory by showing that the exact dimensions of the skewed-T shape are quantitatively determined by the equations of body and surface reflection (such as Torrance-Sparrow). In our new method, we form the color histogram of the pixel values, and measure the dimensions including the ratio of the baseline to the stem height of the T, the angle between the parts, and the position of their intersection. From these measures, by inverting the equations, we can determine the surface roughness and illumination geometry. Since the inversion is mathematically intractable, we use a table lookup. With this method, we have analyzed images of several real plastic objects [Novak and Shafer 92]. This method of analysis is valuable because it is based entirely on the color histogram, and is thus independent of object shape. It also contributes to our basic understanding of the relationship between color, shape, and surface roughness.

Unfortunately, such analysis does assume that the object be reasonably smooth and that the directions of surface normals in the image are widely distributed in all directions. Therefore, it cannot handle cases where only a few planar surface patches exist in an image. To overcome these limitations, we (Sato and Ikeuchi) have developed a novel method for measuring surface and object properties by analyzing a sequence of color images taken with a moving light source [Sato and Ikeuchi 92].

We project the data into a four dimensional space, which we call the temporal-color space, whose

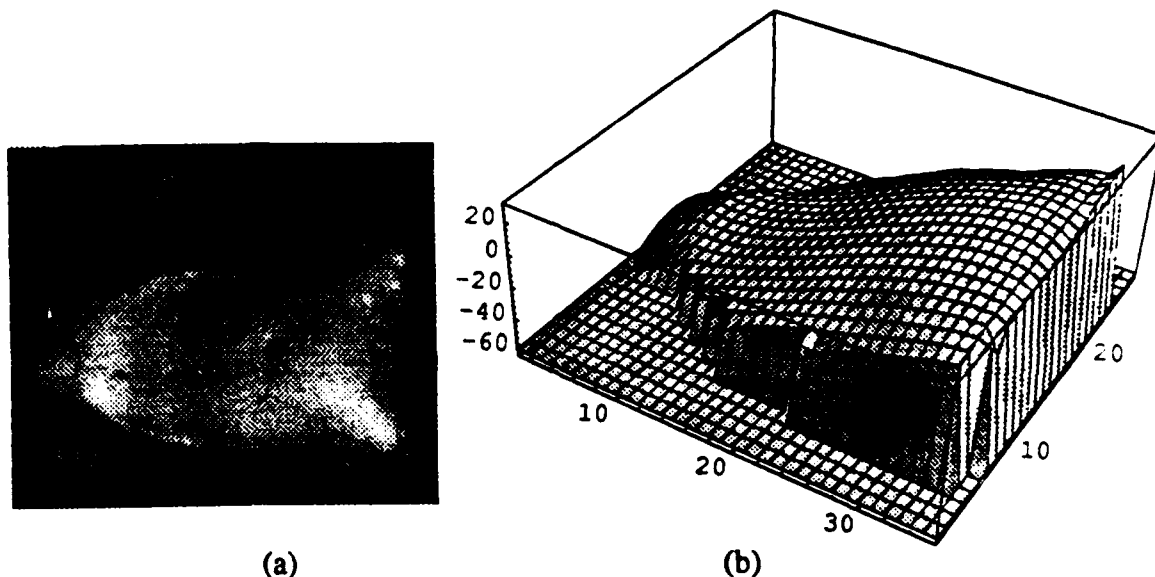


Figure 1: Solder joint measured by photometric stereo: (a) Picture (b) Elevation map

axes are the three color axes (RGB) and one temporal axis. The term "temporal-color space" implies an augmentation of the RGB color space with an additional dimension that varies with time. Because the light source is moving, this dimension represents the geometric relationship between the viewing direction, illumination direction, and surface normal. Thus, many geometries are sampled over time.

The significance of the temporal-color space lies in its ability to represent the change of image color with time, whereas a conventional color space analysis yields the histogram of the colors in an image, only at an instant of time. Conceptually, the two reflection components, surface reflection and body reflection, form two subspaces in the temporal-color space. These two components can be extracted by principal component analysis using the singular value decomposition technique.

This technique has several advantages over other methods for color image analysis: it does not require any prior knowledge about surface reflectance and shape; it can recover the surface orientation and reflectance at each pixel individually; and it does not depend on the assumption of a global distribution of surface normals in the image. This method has been successfully applied to images of real colored objects, resulting in the measurement of the specular reflection component and the body reflection component. These components are subsequently used to recover surface orientation and reflectance at each point, providing an easy-to-implement method for visually localizing and inspecting an object.

1.2. Measuring Shape and Roughness of Metal Surfaces

Metal surfaces are of special interest in physics-based vision because they are important for many manufacturing and inspection tasks, because they exhibit shininess or roughness clearly, and because they are not directly amenable to analysis by color. However, they yield to methods of controlled illumination, provided that the illumination and the reflection model are precisely enough known. Unfortunately, the physics community has primarily chosen to model surfaces that are very smooth, which are important for specialized applications but not for general visual inspection. Therefore, the image understanding community has been forced to develop its own models that are more directly useful for machine vision.

At CMU, we proposed a model in 1991 to unify the well-known models of Torrance-Sparrow, Beckmann-Spizzichino, and Lambert. Later, we built a 3D photosampling device and successfully applied it to measure smooth surfaces such as silicon wafers and transparent plastic lenses. However, since the algorithm used for analysis ignored the specular diffuse lobe component, it could not be applied to rough surfaces such as solder joints or sand-blast finished surfaces, which are commonly found in many industrial parts.

We (Kiuchi and Ikeuchi) have developed a novel algorithm to recover the surface shape and the roughness of such surfaces (Figure 1) [Kiuchi and Ikeuchi 92]. Since the reflectance model depends on surface roughness as well as surface orientation, we are able to recover both shape and roughness

with our method. We take a set of image brightness values measured at each surface point by using our 3D photosampler device. Each brightness value provides one non-linear image irradiance equation, which contains unknown parameters for surface orientation, reflectance, and surface roughness. The algorithm iteratively solves the set of image irradiance equations at each pixel with respect to these parameters. Experiments conducted on several rough surfaces show a high accuracy in estimated surface orientations, and a good estimation of surface roughness. We have been able to determine the shape of a brass cylindrical object, the shape of a solder joint, and the roughness of two nickel microfinish comparators, and we have been able to detect defects in a thick film of gold deposited on a LSI chip package.

2. 3D Shape and Motion Recovery

Geometric methods for recovering surface shape and camera motion are crucial for tasks such as site modeling – the generation of a detailed three-dimensional model of a surveyed site – and for robot vehicle navigation and control. These technologies, in turn, are important for a wide range of military and civilian applications including cartography, reconnaissance, and damage assessment.

These applications require the development of efficient and reliable Image Understanding methods to determine precise three-dimensional shape information from a stationary or moving platform such as a ground or air vehicle, or a stereoscopic camera. We have developed four new approaches to this problem: (1) the image spectrogram for texture and stereo analysis; (2) the multi-baseline stereo method for depth mapping from multiple images; (3) the factorization method for motion analysis under orthography and perspective; and (4) reliably obtaining shape from lens focus and defocus.

2.1. The Image Spectrogram

Image texture is confusing to nearly all vision methods for 3D shape recovery, particularly outdoors. Yet, texture can actually be a rich source of information for measuring surface and terrain shape, classifying vegetation, and defeating camouflage. We (Krumm, Maimone, and Shafer) have developed a new approach for image texture analysis using the image spectrogram, which comprises the "local Fourier transforms" measured in the neighborhood around each pixel. Patterns in the texture of the image are revealed in the structure of each such Fourier transform, and 3D shape is

revealed as a systematic variation from one to the next.

There are two classical problems in image texture analysis: segmenting flat (2D) texture regions in the image, and measuring 3D shape from texture gradients on slanted or curved surfaces. We developed methods for each of these problems that showed the spectrogram can solve each problem as well as other common approaches [Krumm and Shafer 92]. However, the unique power of the spectrogram lies in its ability to integrate these problems in the same representational framework. Using the spectrogram we can solve the combined problem of texture segmentation and shape analysis. The local Fourier transforms of pixels on the same surface are similar, within a linear (affine) transform of each other; but, across surfaces, the Fourier transforms differ significantly. Our analysis proceeds by creating several hypotheses about surface orientation based on small regions in the image. Each hypothesis consists of an estimate of the local surface normal and an estimate of what the local frequency distribution of the texture would look like if viewed frontally. This "frontalized" frequency distribution is computed by undoing the effects of the estimated surface normal on the local frequency distribution. Thus, two adjoining regions of similar texture and different surface normal will have the same hypothesized local frequency distribution. We then merge similar hypotheses to form regions. This method is now being tested in the Calibrated Imaging Laboratory. In addition, we (Maimone and Shafer) are now applying the spectrogram to stereo vision in textured environments, where it appears promising for addressing long-unsolved problems in avoiding false matches and occlusions. The spectrogram is proving to be a powerful bridge between the Fourier-based theories of signal processing and the geometry-based approaches of machine vision.

2.2. Multi-Baseline Stereo

In stereo matching, a longer baseline gives a precise depth estimate, because the depth is calculated by triangulation. A longer baseline, however, poses problems in matching: a longer disparity range must be searched, some parts in the scene may be occluded, and the appearance of some objects in the scene may change significantly between images. Matching becomes more difficult, and there is a greater possibility of false matches. Conversely, a shorter baseline makes matching easier, but reduces the precision of the estimate. There is a trade-off between precision and correctness.

We (Okutomi and Kanade) developed a multiple-baseline stereo technique to solve this problem. This method uses multiple stereo pairs with different baselines generated by a lateral displacement of a camera. Matching is performed simply by computing the sum of squared-difference (SSD) values between multiple stereo pairs. The SSD functions for individual stereo pairs are represented with respect to the inverse distance, and they are simply added to produce the sum of the SSDs. This resulting function is called the SSSD-in-inverse-distance. The range estimate is calculated by finding the minimum of the SSSD-in-inverse-distance curve. This curve shows a unique and clear minimum at the correct matching position even when the underlying intensity patterns of the scene includes ambiguities or repetitive patterns.

Our method has been implemented in software and tested with images from both indoor and outdoor

scenes under a wide variety of conditions [Okutomi and Kanade 92]. Indoors, in a calibrated laboratory, at a distance of 0.5-1.0m; and outdoors, at a distance of approximately 15-35m. We have also tested the method on a large scale outdoor scene shown in Figure 2(a) at the Westinghouse Research Center in Pittsburgh, where Ambler, the CMU Planetary Rover, was tested. The scene contains a grassy field with a line of trees at a distance of 60 m. Six images with horizontal displacements and six additional images with vertical displacement were used. The widest horizontal and vertical baseline in this set was 9cm. Figure 2(b) shows the disparity image. The noisy region is due to lack of features in the area of sky in the original image. This noise is successfully detected by the uncertainty estimate. The plots in Figure 2(c) are the 3D terrain profiles shown as height vs. horizontal distance along the vertical columns drawn in the fig-

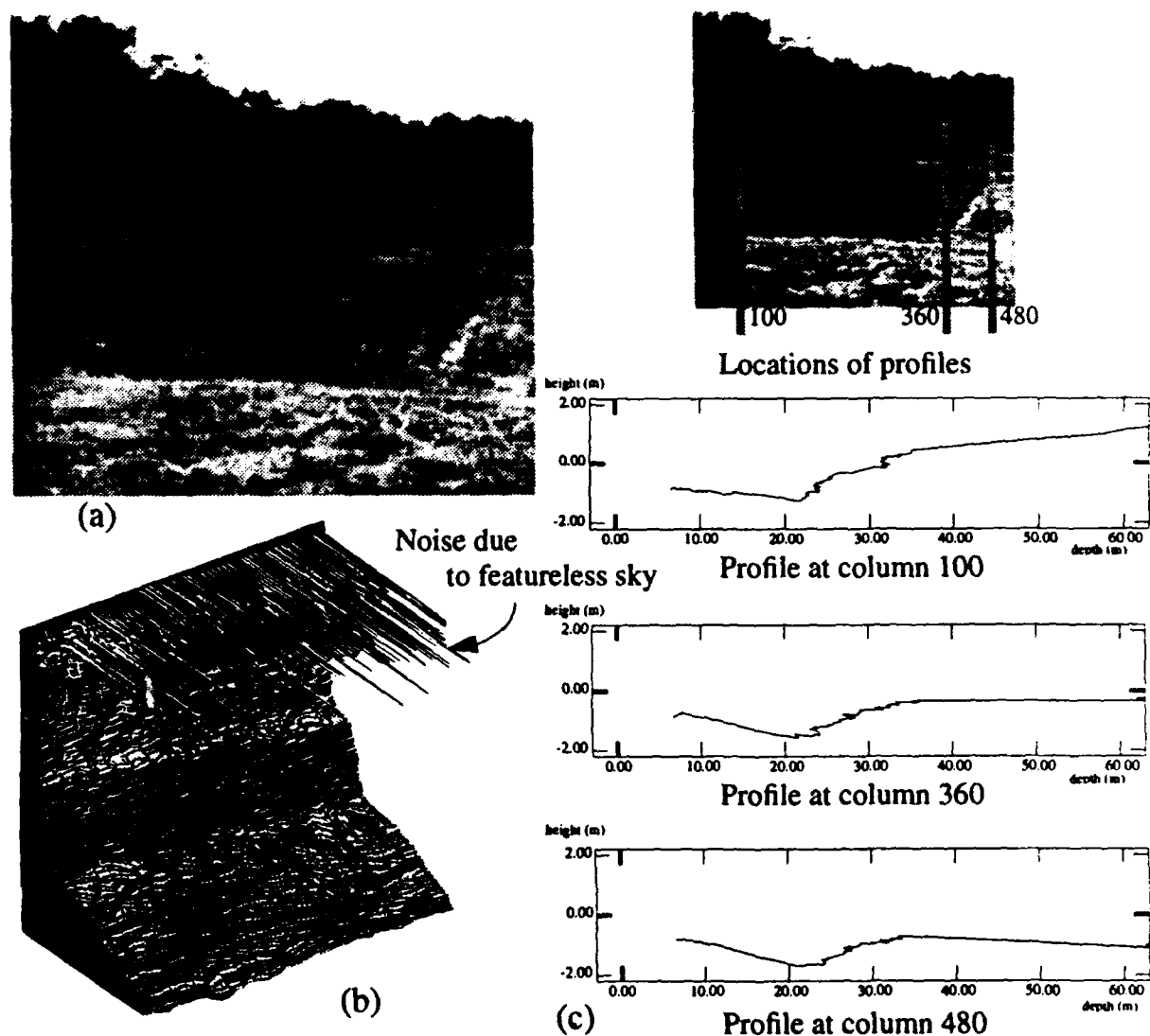


Figure 2: Test site for Multi-Baseline Stereo: (a) Image (b) Depth map (c) Elevation profiles

ure above. We can observe that the terrain features of the scene are correctly recovered; a flat and somewhat descending region at the front, a slope in the middle, and then a more gentle slope at the rear before the tree line. The measured distances to a few points in the scene have been verified to be correct to within 1%.

The multi-baseline stereo system has been put to practical use on a CMU robot named Dante, a large, 8-legged walking machine sponsored by NASA and NSF for the exploration of a live volcano in Antarctica. The system uses three cameras arranged along a 1-meter horizontal baseline. The system software was highly optimized in favor of speed in exchange for somewhat reduced precision. Output is a dense depth map of 256x256 pixels with 40 disparity levels in 7 seconds, which enables the robot to move slowly (2-3 MPH) through a field of obstacles. The system has also been used to guide the CMU robotic truck, NAV-LAB II.

2.3. Factorization for Motion Analysis

Recovery of 3D shape and camera motion information is important for robot vehicle control and terrain mapping. Unfortunately, the problem can be mathematically unstable. We developed a novel factorization method for analyzing image sequences that avoided the instability by using singular value decomposition of the observation data. The method was limited to telephoto lenses due to its use of the orthographic projection model.

We (Poelman and Kanade) have recently developed a new factorization method that uses a paraperspective projection model [Poelman and Kanade 92]. Paraperspective correctly models several aspects of real camera image projection which orthography fails to account for, including the change in the image size of an object as it moves towards or away from the camera, and the changing angle from which an object is viewed as the object translates across the field of view. These properties allow the method to be used in a much wider range of situations, and allow the recovery of the distance from the camera to the object. Yet the paraperspective projection model, like orthographic projection, can be described by linear equations. This allows us to recover the shape and motion in an efficient and robust manner similar to the original factorization method. The method has been tested on both synthetic and real data, and will enable reliable terrain mapping from a moving vehicle with a single camera.

2.4. Camera and Lens Modeling

Although the basic optics of lenses and cameras is known in theory, most models do not account for the aberrations encountered in practice. At CMU, in the Calibrated Imaging Laboratory, we (Willson, Xiong, and Shafer) are developing new models for modeling, calibration, and control of automated zoom lenses. Our recent work has led to advances in the calibration of image center and in developing 3D shape recovery from lens focus.

Nearly all methods for machine vision assume that the image center is considered to be the point of intersection of the camera's optical axis with the camera's sensing plane. In fact, there are many possible definitions of image center, and in real lenses most do not have the same coordinates. We have identified 16 different ways to define "image center", and developed a taxonomy of image centers based on the number of different camera settings used and on the type of measurements that are made during calibration [Willson and Shafer 93]. By using the proper image center for each image property that we are trying to model and by calibrating the image centers over the appropriate ranges of lens parameters, we have improved the precision of a standard (Tsai-Lenz) calibration from 0.23 ± 0.10 pixels RMS error to 0.06 ± 0.04 pixels.

Based on our new lens models, we have developed new methods that dramatically improve 3D shape recovery from lens focus and defocus [Xiong and Shafer 93]. In the range-from-focus task, we obtain an accuracy of 1 part in 1000 at distances of 1.2m, which is an improvement by fivefold over previously published results for this task. The improvement comes from smoothing the criterion function fitting a polynomial curve to it in the vicinity of the peak value, where noise becomes the limiting factor to precision. More significant, for range-from-defocus, we made two improvements – we use an iterative method to overcome the effects of window size on the calculation, and we developed an improved blur model in terms of motor control variables rather than abstract optical idealizations, allowing more accurate calibration. Taking just two images with differing focus, we have obtained dense depth maps with a precision better than 1 part in 200, compared to results of 1 part in 75 reported in the literature. With this precision, depth-from-defocus is becoming a viable complement to more established techniques for 3D shape recovery such as stereo vision.

3. Computational VLSI Range Sensor

While vision software is useful for long-range 3D shape recovery, at short distances, hardware can be used to accomplish the purpose more rapidly and reliably. This requires computational sensors that place computing power on the sensing chip itself to perform the range computation without the bottleneck of data transmission to a CPU. The resulting sensors will be useful for robotic inspection, manipulation, and control in real time.

At CMU, we (Gruss, Tada, and Kanade) have developed such a VLSI range-image sensor based on the light-stripe triangulation technique, which has proven to be robust as well as amenable to hardware implementation. Our sensor (Figure 3) produces 100 frames of range data per second, which is two orders of magnitude faster than conventional light-stripe sensors [Gruss et al. 1992].

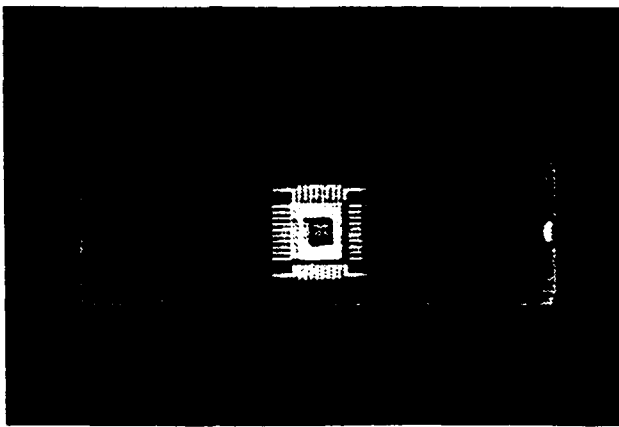


Figure 3: This chip measures a 32x32 image of range data, 1000 times per second

The chip consists of an array of photosensitive cells which independently determine when they see light from the stripe reflected back by objects in the scene. Working in parallel, the 32x32 array of cells acquires a 1,024 pixel range image in one millisecond. The accuracy and repeatability of each pixel has been measured to be within 0.5 mm at 500 mm distances (0.1%).

This sensor, the second version of our design, has cells that are 40% smaller than our first design, giving an increase from 28x32 to 32x32 cells. In addition, a true peak detector has replaced the thresholding circuit previously used to identify the light stripe. The new peak detection scheme has two important advantages. First, the new design is more sensitive to the stripe, which allows the sensor to operate in the presence of bright indoor

ambient lighting, and the increased sensitivity permits its use on a wider variety of objects. In addition, the range sensor chip now provides reflectance information as an artifact of the new peak detection process. The reflectance image is read from the chip along with acquired range data, and pixels of the reflectance image are perfectly aligned with corresponding pixels in the range image. The reflectance data is useful for object recognition, and we also use it for efficient calibration of the VLSI sensor itself.

One of the distinguishing features of this research is the very practical nature of the problem that has been solved – our sensor technology provides the high frame rates required by the most demanding autonomous robotic systems. The advantages of VLSI computational sensing have been advocated by many, but few practical sensors of this type have been developed. We now plan to deploy these systems for research in robotic applications within the DARPA and NSF communities.

4. Parallel Vision

In addition to sensor hardware, researchers at CMU are using parallel computing to speed up vision software for practical application. One focus of our parallel vision efforts (Webb) has been the development of languages for easy but efficient programming of image processing operations. This led to Adapt, which covers both local and global operations. Recently, Adapt was released commercially by Intel Corporation for the iWarp computer. Plans are in place to support Adapt on future parallel computers from Intel, as part of the effort to support current iWarp users. Based on our experience with Adept, we have developed a new parallel FORTRAN with a "Do&Merge" loop construct which allows the programmer to describe a parallel program at two levels (similar to Adapt), one describing an operation to be performed in parallel across an iteration range, and another describing how to combine the independently computed results. This Do&Merge loop is being incorporated into the CMU FORTRAN compiler for easy generation of efficient image processing programs.

The Adapt language has been used to develop an efficient implementation of Kanade-Okutomi multi-baseline stereo vision [Webb 93]. The iWarp implementation of Adapt allows processing of three 240x256 images to recover sixteen disparity levels. Plans are underway to demonstrate real-time performance by using an Ironics framegrabber for the camera interface; we expect to process

10 stereo pairs per second. This passive 3D vision system will be tremendously useful in a wide variety of robotic applications, including SSV, UGV, Air Vehicle, and industrial applications.

5. Vision for Object Recognition and Manipulation

Robotic manipulation of objects is one of the most important application of autonomous systems. At CMU, we have developed a new representation for recognizing curved objects, expanded our ability to automatically generate vision programs with the Vision Algorithm Compiler (VAC), and learning assembly by observing a human do the task.

5.1. Surface Modeling for Recognition

Recognizing curved objects is important for manipulating and inspecting equipment, vehicle parts, and manufactured items. CMU researchers (Delingette, Hebert, and Ikeuchi) have developed a novel representation of curved objects called the Simplex Angle Image that allows matching even when one of the objects is partly blocked from view [Delingette et al. 93].

To compute the SAI, we begin with dense 3D range data. We pose an ellipsoidal mesh of points that surrounds the object, and by a method based on deformable surfaces, we shrink the mesh to fit the surface data closely and in a way that is unique, regardless of the object's angle of orientation. At each node on this mesh, we compute the "simplex angle" which expresses the 3D curvature of the surface between this node and its neighbors on the mesh. Each simplex angle is paired with the surface normal at that point, and mapped onto the point of the unit sphere corresponding to the normal. The result, called the Simplex Angle Image (SAI), is a spherical representation of the object. The key feature of this representation is that two instances of the same object in two different poses have the same SAI up to a rotation of the unit sphere. Using this approach, we have demonstrated the recognition of 3-D curved objects in range images of complex scenes with multiple objects, and we have also used the SAI to piece together multiple views of a complex 3D object such as a hand.

5.2. Vision Algorithm Compiler

The Vision Algorithm Compiler (VAC) is a method developed at CMU to replace the expensive custom-building of vision software with an automated

vision programming system. Provided with symbolic descriptions of the objects, sensor, and task, the VAC generates a vision program that can be rapidly executed on-line to perform the task. This approach will greatly reduce the cost of constructing vision systems for robotic manipulation and assembly. To prepare the VAC for deployment, we have made recent advances in two problem areas: planning multiple observations to resolve ambiguities, and developing an efficient algorithm for object recognition for large object databases.

We (Ikeuchi, Wheeler and Gremban) have developed techniques to utilize sensor motions to acquire observations to resolve ambiguous interpretations of an object's pose. The solution utilizes a resolution tree specifying the motions from the initial vantage point that will reduce the ambiguity until a unique pose can be resolved from the observations. The resolution tree is created off-line by a traditional planner that utilizes knowledge of the aspect classes and the spatial extent of the aspects. New data representations were developed to facilitate this planning operation. We have conducted experiments in recognizing specular objects and finger-gap sensing, showing the utility of multiple observations for resolving pose ambiguity.

We have developed a novel algorithm for object recognition in range images that is efficient for large model databases [Wheeler and Ikeuchi 93]. Our algorithm performs optimal selection of hypothesized views to model, and for each one, the visible image features are generated through simulation of the imaging and feature extraction process. In these simulated views, the correspondence of model features and image features is known. Using this correspondences, statistics are accumulated to produce conditional probability distributions of the extracted features given the visibility of each model feature in the scene using Markov Random Fields and a Highest Confidence First estimation technique. This method has proven effective in substantially reducing the number of hypotheses to be verified while choosing accurate hypotheses. Because of ability to search through large sets of possible hypotheses, this method allows us to automatically generate vision programs for large sets of objects and also for partly occluded objects.

5.3. Learning From Observation

We (Ikeuchi, Kang, Suehiro, Kawade) have been working on a task programming approach called Assembly Plan from Observation (APO), which will enable the robot system to observe a human

perform a task, understand it, and perform the same task with minimum human intervention. In this approach, the human provides the intelligence in choosing the initial hand (end-effector) trajectory, the grasping strategy, and the hand-object interaction by directly acting them out. This approach helps to alleviate the problems of symbolic path planning, grasp synthesis, and task specification.

Recently, we have developed a method of APO for objects with curved surfaces [Ikeuchi et al. 93]. Each surface patch is categorized according to the signs of its Gaussian and mean curvatures. In our current implementation, the human operator demonstrates a task one step at a time to the system. Each task is captured in intensity and range images. The intensity images (sampled at a regular interval) are analyzed to detect the next meaningful action of the human operator, while the range images are used to recognize objects and locate the hand in the scene. A significant brightness difference between consecutive intensity images signals the occurrence of a meaningful action, and triggers the range finder to capture the range image of the scene. The system roughly locates the grasping points from the last two images by image subtraction, superquadric fitting of the distal portions of the thumb and index finger, and determination of the intersection points between the superquadrics and the grasped object. The contact transition is recognized based on the before- and after-task range images; the appropriate task model is determined from the contact relationship (Figure 4) and then instantiated with the relevant motion param-

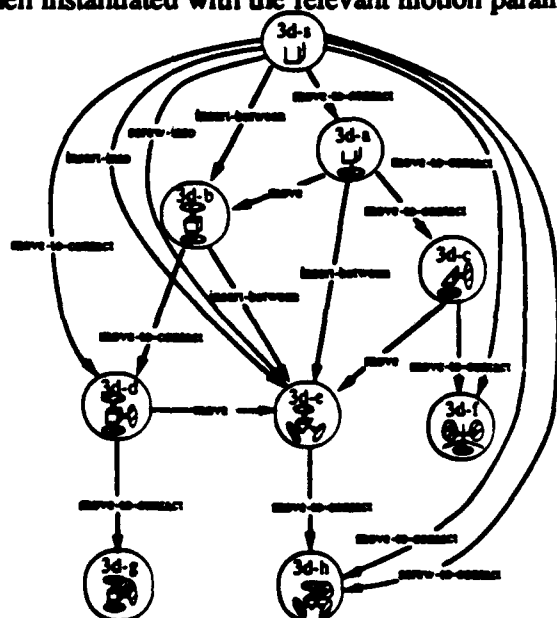


Figure 4: Using this table, the needed motion is determined from "before" and "after" contacts

ters. This, coupled with mechanical properties such as bolt-like and nut-like motions, enables the robot to replicate tasks such as picking and placing, and screwing a bolt into a hole.

Grasp recognition is central to automatic learning of a grasping task. We (Kang and Ikeuchi) have developed a representation called the contact web in conjunction with a grasp taxonomy to identify a grasp (Figure 5) [Kang and Ikeuchi 92]. The grasp is represented by a grasp abstraction hierarchy which comprises high-level (type of grasp), intermediate-level (finger groups), and low-level (locations and joint angles) information.

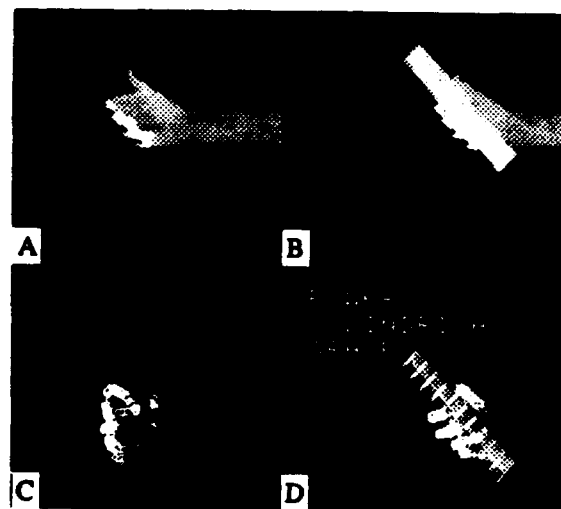


Figure 5: Recognizing a grasp (a) Range image of hand (b) Hand + object (c) Solid model (d) Grasp

Our method for grasp recognition detects all phases of the grasping operation. Given a temporal image sequence of a grasping task, the pregrasp phase is first inferred approximately (using parameters such as speed, grip aperture, and approach polygon area) until the grasp itself has been temporally located in the sequence and identified. Preliminary work has indicated that we can temporally locate the static grasp phase within the sequence by analyzing both the speed and approach polygon area profiles. The identification of the grasp will either strengthen or weaken the pregrasp phase hypothesis. In addition, it constrains the types of manipulation that can be applied to the object. Our system for recognizing grasping tasks comprises a CyberGlove hand tracker (with a Polhemus device), a monochrome camera, and a range finder.

6. Vision for Robot Vehicles

Mobile robots are vital for reconnaissance, exploration, and all missions to be carried out in remote locations. Navigation by GPS and dead reckoning can be used to tell where the vehicle is, but to guide it through terrain or to reach a target, visual sensing is needed. This is challenging because of the need for reliability in a complex, ever-changing outdoor environment. CMU has been a leader in vision-guided robot vehicle development, and has several major vehicle programs: the Navlab/UGV effort for wheeled land vehicles, the Ambler for legged locomotion in rough terrain, and a new autonomous helicopter.

6.1. Navlab and UGV

CMU is a key member of the DARPA Unmanned Ground Vehicle (UGV) program. Based on our Navlab autonomous van and our Navlab II HMMWV, we are providing basic mobility for the Demo II effort, including perception, planning, vehicle control and modeling, and human-computer interaction. We have recently demonstrated cross-country navigation for 5 km; autonomous parallel parking (including finding the parking space); and a mission that includes driving on dirt roads, avoiding obstacles, following a map, driving off road, and stopping at a designated landmark. Our software is being transferred to Martin Marietta for integration into the UGV testbed vehicle.

We have made advances in several areas to achieve these goals:

Multiple types of roadway: ALVINN (Pomerleau) is our neural-net based road following program, which learns to drive from five minutes of observing the human driver's control of the vehicle. Recently, we have extended ALVINN to drive on a wide variety of roads, using information from several different networks, each trained for an indi-

vidual road type (Figure 6) [Pomerleau 92]. Different networks are selected based on the match between the input scene and its encoding as represented in each network. MANIAC (Jochem, Pomerleau, and Thorpe) does not select an individual network, but instead uses an additional neural network which looks at the outputs of the individual nets [Jochem et al. 93]. This top-level net can then use input from each of the lower-level nets to produce a steering response which may be superior to any of the responses from the individual nets.

Obstacle location and avoidance: GANESHA (Langer, Hebert, and Thorpe) uses several sonar sensors placed around the front of the vehicle for obstacle avoidance using an occupancy grid representation centered on the moving vehicle. It has also been used for parallel parking (Figure 7) [Langer and Thorpe 92], and the moving occupancy grid representation has been used to integrate stereo vision and laser range data as well.

Intersections: YARF (Kluge and Thorpe), our system for driving by tracking lane markings, can now detect and navigate intersections as well as roadway stretches [Kluge and Thorpe 93]. This will allow autonomous traversal of road networks.

Teleoperation: STRIPE (Kay and Thorpe) is a method of semi-autonomous teleoperation of a vehicle which allows it to accurately traverse hilly terrain while communicating with the operator across a very low bandwidth link [Kay and Thorpe 93]. The operator plots the vehicle's chosen trajectory based on a single 2-D image, and the transformation of 2-D image points to 3-D world points is done in real time on the vehicle.

Automatic convoying: RACCOON (Sukthankar) is a vision system that tracks taillights for car-following at night, building a map in real time of the lead vehicle's position for accurate control [Sukthankar 92]. It has been demonstrated on the NAVLAB II at 32 km/h on a winding road.

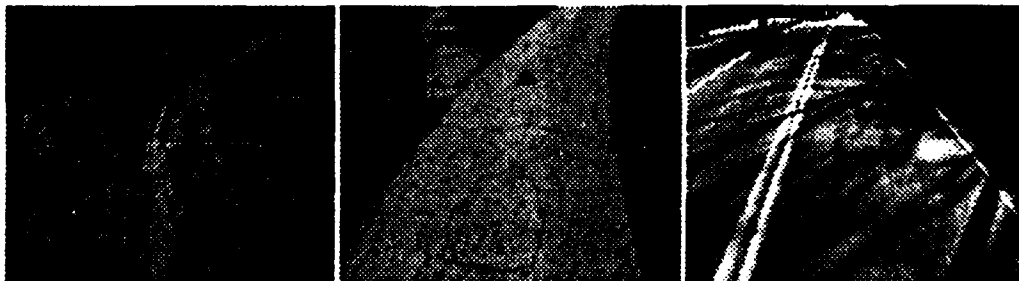


Figure 6: ALVINN has been trained for several roadways: dirt road, bicycle path, two-lane highway

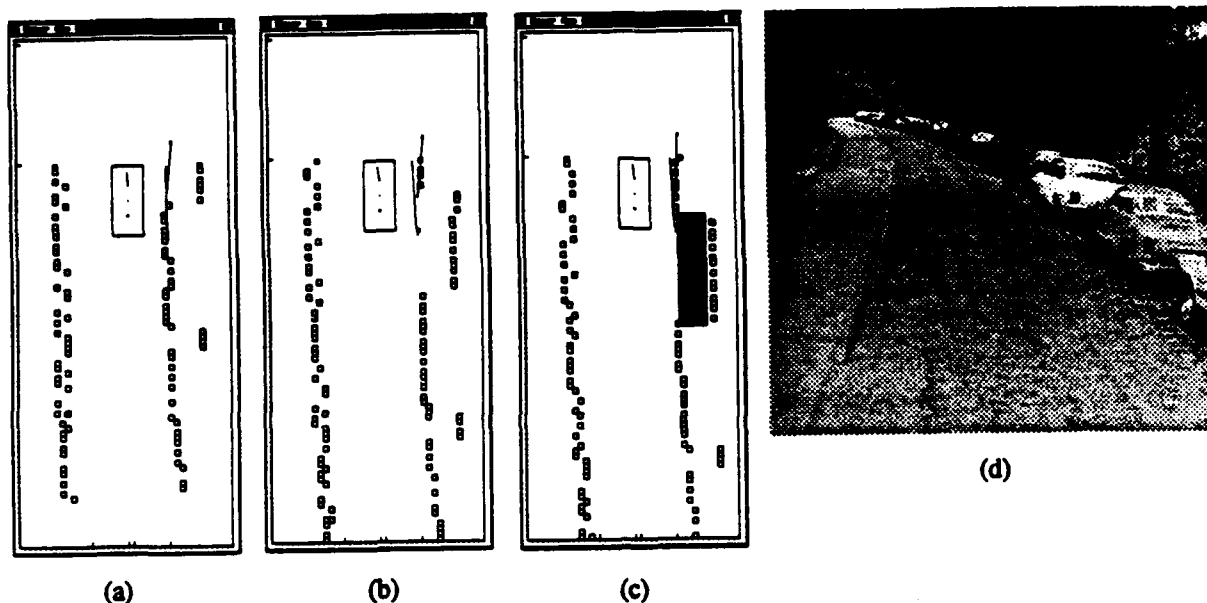


Figure 7: Automated parallel parking: (a) Approach (b) Detect gap (c) Prepare to enter (d) Roadway

Landmark recognition: Landmark recognition is also important for both roadway and off-road navigation, and we (Hebert) have a new approach based on surface matching with range data that has been integrated on the Navlab to register the vehicle position with the map [Hebert 92].

6.2. Ambler for Planetary Exploration

In its fourth year of operation, the Ambler walking robot established new world records for long-term autonomous walking, both outdoors and indoors. With all computing, sensing, power, and telemetry on-board, the robot is completely self-reliant. To date, the Ambler has walked autonomously a total of over 4 km, much of it over rugged, difficult terrain.

We (Krotkov) have extended the Ambler terrain mapping system to operate reliably and continuously under a wide variety of environmental conditions in natural, outdoor environments [Krotkov et al. 93]. The fielded system has been thoroughly tested in numerous walking experiments, processing tens of thousands of range images and tens of millions of terrain elevation points. In a single run, the perception system acquired 1200 range images and built 4700 terrain elevation maps containing a total of 2.6 million elevation points.

The extensions include using feedback from leg contact with the terrain to increase the accuracy of the elevation maps, aggressive preprocessing of

range sensor data, periodic recalibrations to compensate for long-term sensor drift, and extensive error detection and recovery procedures to respond to hardware and memory management errors. We also developed a new fractal-based method for recovering the terrain map from range data.

6.3. Autonomous Vision-Guided Helicopter

We (Amidi and Kanade) are also developing a vision-guided autonomous helicopter. At present, we are researching visual feedback for close and precise helicopter hovering near a known object of interest to perform inspection tasks. Our control scheme is based on a linear helicopter control model which is updated in real-time by a fuzzy controller. The helicopter model is the basis for image feature detection and tracking as well as helicopter control. We have been testing our control ideas on a model electric helicopter using an indoor 6-degrees-of-freedom (6-DOF) testbed that provides both ground truth data and controlled test environments. We plan to upgrade to a mid-size helicopter (Yamaha R50, 2.6-meter long, 20 kg payload) with a larger 6-DOF stand for both indoor and outdoor experiments. Finally, we plan to perform free flight experiments using a camera and on-board sensors to fly the helicopter while performing a real-world task.

7. Vision for Human-Computer Interaction

It is now being recognized by DARPA and others that one of the biggest barriers to the effective application of computer technology is the difficulty of communicating between the human and the computer. The Image Understanding group at CMU has found several ways that machine vision can contribute to improving this communication to impact all uses of computers, from word and data processing to gesture input and even for improving the study of Human-Computer Interaction (HCI) itself. Our current focus areas are in gesture input, perception of the user's face, tracking where the user is looking, and aids for surgery.

7.1. The Vision Dataglove

Gesture input is useful for tasks such as robotic manipulation, directing the attention of a robot vehicle, and providing input for map-based mission planning and coordination. We (Rehg and Kanade) are developing the "Vision Dataglove", is a system for model-based visual tracking of human arm and hand motion. By exploiting geometric and kinematic models of the human hand and arm, we can estimate its motion from a sequence of intensity images. We model the hand as a collection of 16 rigid segments (12 individual finger segments, 3 thumb segments and one segment for the palm). The vision dataglove has potential applications in man-machine interfaces and teleoperation.

7.2. Face Perception

The second focus of our work on vision for HCI is in perception of the human face. We (Rander and Kanade) have demonstrated a system for tracking specific facial features such as the eyes, nose and mouth. The system builds a multi-resolution image pyramid from a digitized image of a person's face. It uses coarse templates to localize the person's face within the lowest resolution image in the pyramid. It then searches the corresponding region of the higher resolution images for smaller facial features such as the eyes, nose and mouth. Constraints imposed by the positions of features in the previous image and by a geometric model of facial feature relationships allows the system to limit its search and achieve near real-time cycle rates (currently about 5 Hz).

7.3. Tracking the User's Gaze

One of the ways machine vision can aid HCI is by

providing a hands-free replacement for a pointing device. One such system we (Pomerleau and Baluja) are developing is a non-intrusive gaze tracker based on artificial neural networks. Once the position of the eyes are located in the video image, the gaze tracker extracts a small window centered on the right eye of the person, and provides it as input to a neural network. The network is trained to determine where on the computer screen the person is looking from the appearance of his eye in the input image. By exploiting the unique characteristics of an individual's eye appearance, it is able to estimate the location of a person's gaze to within approximately 1 degree (about the size of a 6 letter word viewed in a normal font from a comfortable distance from the screen). This level of accuracy is comparable to that of the most expensive commercially available vision-based eye trackers. Unlike conventional eye-trackers, it does not require the user's head to be fixed in a frame – the user simply sits normally in a chair. The gaze tracker is useful as a rapid, hands-off pointing device, and also for studying the process of human-computer interaction itself, to improve interface designs.

7.4. Vision-Aided Laparoscopic Surgery

Laparoscopic surgery is a minimally invasive surgical technique which involves low trauma, reduced risk of infection, and less post-operative pain than conventional open surgery. Unlike open surgery of the digestive system, which involves a large incision, laparoscopy is performed by special instruments inserted through small holes cut into the abdomen. The interior of the patient is imaged by a small camera mounted on a special instrument, called a laparoscope, and displayed on a standard video monitor.

We (Gibson and Kanade) are addressing two of the fundamental limitations of current laparoscopic surgery: (1) The laparoscope camera does not provide three-dimensional depth perception, and (2) misalignment of the camera view can greatly increase the complexity of hand-eye coordination for the surgeon. We are developing a prototype system that uses a pair of video cameras to image the operative field. This stereo view will allow the surgeon to perceive three-dimensional depth. Electronic sensors will be fixed to both the laparoscope and the surgeon to help align the camera view.

References

[Delingette et al. 93]

H. Delingette, M. Hebert, and K. Ikeuchi. A

- Spherical Representation for the Recognition of Curved Objects*. Proc. ICCV-93, Berlin, May 1993, to appear. Short version in these proceedings.
- [Gruss et al. 92]
A. Gruss, S. Tada and T. Kanade, *A VLSI Smart Sensor for Fast Range Imaging*, IROS 92, Raleigh, NC, July 1992. In these proceedings.
- [Hebert 92]
M. Hebert. *3D Landmark Recognition from Range Images*. Proc. CVPR-92, Urbana-Champaign, June 1992, p.360-366.
- [Ikeuchi et al. 93]
K. Ikeuchi, M. Kawade, and T. Suehiro. *Towards an Assembly Plan from Observation, part III: Assembly Task Recognition (General Case)*. ICRA, May 1993, to appear.
- [Jochem et al. 93]
T. Jochem, D. Pomerleau, and C. Thorpe, *MANIAC: A Next Generation Neurally-Based Autonomous Road Follower*, IAS-3, Pittsburgh, PA, February, 1993. In these proceedings.
- [Kang and Ikeuchi 92]
S. B. Kang and K. Ikeuchi, *Grasp Recognition Using the Contact Web*. IROS 92, Raleigh, NC, July 1992, pp. 194-201.
- [Kay and Thorpe 93]
J. Kay and C. Thorpe, *Supervised Telerobotics Using Incremental Flat-Earth Geometry: STRIFE*, IAS-3, Pittsburgh, PA, February, 1993, to appear.
- [Kiuchi and Ikeuchi 92]
T. Kiuchi and K. Ikeuchi. *Determining Surface Roughness and Shape of Specular Diffuse Lobe Objects Using Photometric Sampling Device*. IAPR-92 Workshop on Machine Vision Applications, December 1992, Tokyo.
- [Kluge and Thorpe 93]
K. Kluge and C. Thorpe, *Intersection Detection in the YARF Road Following System* IAS-3, Pittsburgh, PA, February, 1993, to appear.
- [Krotkov et al. 93]
Krotkov, E. and Simmons, R. and Whittaker, W., *Autonomous Walking Results with the Ambler Hexapod Planetary Rover*, IAS-3, Pittsburgh, PA, February, 1993, to appear.
- [Krumm and Shafer 92]
Krumm, John and Steven A. Shafer, *Shape from Periodic Texture Using the Spectrogram*, CVPR-92, pp. 284-28, Also appeared as report CMU-RI-91-29, 1992.
- [Langer and Thorpe 92]
D. Langer and C. Thorpe, *Sonar-based Outdoor Vehicle Navigation and Collision Avoidance*. IROS 92, Raleigh, NC, 1992.
- [Novak and Shafer 92]
C. L. Novak and S. A. Shafer, *Estimating Scene Properties from Color Histograms*, Technical report CMU CS-92-212, November 1992. Short version in these proceedings.
- [Okutomi and Kanade 92]
M. Okutomi and T. Kanade, *A Locally Adaptive Window for Signal Matching*, International Journal of Computer Vision, Vol. 7, No. 2, pp. 143-162, 1992.
- [Poelman and Kanade 92]
C. Poelman and T. Kanade, *A Paraperspective Factorization Method for Shape and Motion Recovery*, Technical report CMU-CS-92-208, October 1992. In these proceedings.
- [Pomerleau 92]
D. Pomerleau, *Knowledge-based Training of Artificial Neural Networks for Autonomous Robot Driving*, In *Robot Learning*, J. Connell and S. Mahadevan, eds. 1992
- [Sato and Ikeuchi 92]
Y. Sato and K. Ikeuchi, *Temporal-Color Space Analysis of Reflection*, Technical report CMU-CS-92-207, November 1992.
- [Webb 93]
J. Webb. *Implementation and Performance of Fast Parallel Multi-Baseline Stereo Vision*. In these proceedings.
- [Wheeler and Ikeuchi 93]
M. Wheeler and K. Ikeuchi. *Sensor Modeling, Markov Random Fields, and Robust Localization for Recognizing Partially Occluded Objects*. In these proceedings.
- [Willson and Shafer 93]
R. Willson and S. Shafer. *What is the Center of the Image?*, Submitted to CVPR-93.
- [Xiong, and Shafer 93]
Y. Xiong and S. Shafer, *Depth From Focusing and Defocusing*, submitted to CVPR-93. Short version in these proceedings.

Progress in Computer Vision at the University of Massachusetts

Allen R. Hanson, Edward M. Riseman, and Charles A. Weems

**Computer Vision Research Laboratory
Dept. of Computer and Information Science
University of Massachusetts
Amherst, MA 01003**

Abstract¹

This report summarizes progress in image understanding research at the University of Massachusetts over the past year. Many of the individual efforts discussed in this paper are further developed in other papers in this proceedings. The summary is organized into several areas:

1. Mobile Robot Navigation
2. Motion Analysis
3. Interpretation of Static Scenes
4. Image Understanding Architecture
5. RADIUS Image Exploitation

The research program in computer vision at UMass has as one of its goals the integration of a diverse set of research efforts into a system that is ultimately intended to achieve real-time image interpretation in a variety of vision applications.

1. Mobile Robot Navigation

1.1. Automated Model Acquisition and Extension

The focus of the UMass mobile robot navigation project is robust landmark-based navigation, with a focus on automated model acquisition and model extension. Thus, for navigation in unmodelled or sparsely modelled environments, our general scenario would involve the initial acquisition of prominent visual features that can serve as landmarks. This initial phase of partial model acquisition is necessary because there are few situations where a model of a complex outdoor scene will be available a priori. Once a sparse model is available, then the vehicle position and orientation (i.e. pose) can be recovered by recognizing landmarks. The model extension

phase involves tracking new unmodelled features (points and/or lines), and using the landmarks and partial model to determine the camera pose for triangulation of the new features and incorporation into the 3D model.

Most of the algorithms have been described in previous IUW proceedings and the general vision literature [Beveridge 92, Kumar 92, Sawhney 92, 93]. These algorithms have been shown to be very accurate in many indoor experiments using a camera mounted on a mobile robot and on a moving robot arm. One new experiment that integrated several components involved the detection of shallow structures - an aggregation of line features that can be approximated in an image sequence as a frontal planar surface. The 3D position of these features served as the acquired model, with a depth error of less than 4%. As motion of the camera continues, the model is extended with depth information on other tracked points to accuracies of less than 2% error in depth.

1.2. Status of the UMass Mobile Perception Laboratory (MPL)

1.2.1. Physical Description

The UMass Mobile Perception Laboratory (MPL) is based on a significantly modified HMMWV. The design of the overall system includes actuators and encoders for the throttle, steering column and brakes that closely match those being used by CMU, controlled by 68020's in a 6u VME cage. The low-level control software for controlling speed and steering angle will also be the same as that of CMU. The modifications and component installation is being performed by RedZone, Inc., a Pittsburgh-based firm specializing in custom robotics, and was completed at the beginning of February 1993.

Electrical power is supplied by a 10kW diesel generator, whose output is split into two 5kW circuits. The first circuit is conditioned and backed by a 5kW uninterruptible power supply (UPS) system, and is used to supply power to all sensitive electronic equipment. The second circuit

¹This research has been supported in part by the Defense Advanced Research Projects Agency under TACOM contract number DAAE07-91-C-R035, HDL contract number DAAL02-91-K-0047, and TEC contract number DACA76-92-C-0041, by the National Science Foundation under grant CDA-8922572, IRI-9113690, and IRI-9208920, and by RADAC under contract number F30602-91-C-0037.

is not conditioned and is used to power the air conditioners. Both circuits are attached to a shore-power hook-up that provide an alternative power source to the on-board generator.

The physical lay-out of equipment was designed to

- 1) provide for two on-board programmer stations,
- 2) minimize destructive modifications to the body of the vehicle, and
- 3) keep the center of gravity as far forward as possible, in order to minimize stress on the suspension system.

The first programmer station is located in the HMMWV's passenger seat, with a 17" color x-terminal fixed to the metal platform between the passenger's and driver's seats. The second programmer station is located behind and slightly above the driver, and includes a car seat, mounting brackets for both an SGI color terminal and a small SONY monitor for viewing raw TV signals.

The back of the vehicle is filled with equipment. On the driver's side of the vehicle, behind the second programmer station, is all equipment associated with providing power. On the passenger's side there are four enclosed, air conditioned 19" computer frames for the on-board computer systems. The first frame will hold the 6u VME cage for throttle, brake and steering controllers and a second 6u VME cage for holding digitizers, image frame stores and a Datacube MaxVideo20. The second computing frame will contain a 9u cage for the Silicon Graphics four-node multiprocessor, as well as the SGI's disk drives, power supply and (removable) tape drive. The third frame is reserved for the Image Understanding Architecture (IUA). The fourth frame is for future additions, including video recorders for collecting data and recording experiments. Together, the four frames take up the length of the vehicle's bed, as do the programmer station, UPS cage and generator on the left side.

1.2.2. Sensor Configuration

The vehicle's sensor package includes a Stagat, which is a rotating stabilized platform being supplied to the UMass and CMU vehicles by TACOM. The UMass Stagat is mounted on a level platform located at the center of the roof of the cab. We are planning to put two CCD color cameras on the Stagat, one with a wide angle lens and the other with a telephoto lens. The first will be used to locate landmarks in the larger scene, and the second will be used for landmark matching and accurate pose refinement. The Stagat will also contain a FLIR sensor. The Stagat's hardware is mounted above the driver's head in the enclosure

originally occupied by the HMMWV's NBC system. Forward of the Stagat, at the edge of the cab's roof, is a long (5" by 12" by 12") rectangular enclosure with a glass front and hinged roof for forward-looking stereo cameras.

1.2.3. Software Environment

MPL is an experimental laboratory for testing and integrating different approaches to problems in autonomous navigation, including, but not limited to, landmark-based navigation, obstacle detection and avoidance, model acquisition, and road following. It is therefore important that MPL have a software environment where multiple visual modules, addressing different subtasks, can be easily integrated, and where researchers can quickly experiment with different combinations and parameterizations of those modules. At the same time, MPL's software environment must be efficient enough to meet the demands of real-time navigation research.

The need to balance between flexibility and efficiency has led us to design a software environment with two major components: the ISR3 in-memory data store, and a graphical programming interface adapted from Khoros. ISR3 is the glue that binds independent visual modules together [Draper 93a]. It is an in-memory database that allows users to define structures for storing visual data, such as images, lines and surfaces. ISR3 then serves as a buffer, so that, for example, lines produced by one module can be used by another, even if the second module is run later or on a different processor than the first. ISR3 also provides modules with efficient spatial access routines for visual data, and protects data from being simultaneously modified by two or more concurrent processes. The graphical programming interface allows programmers to easily sequence modules and modify their parameters.

1.2.4. Navigation System

A preliminary version of a behaviour-based system for determining vehicle pose from known landmarks has been designed. It is assumed that pose estimates and associated covariance (error) estimates are returned from several subsystems (GPS, INS, Landmarks, and dead reckoning) asynchronously. These estimates are continually combined via a Kalman filter into a single pose estimate (and associated covariance matrix estimate) and stored in a vehicle state vector. The vehicle pose error is continually monitored in a simple loop which branches to a behavior selection strategy when the vehicle pose error exceeds a preset threshold.

The system also contains a video image frame buffer and STAGET control subsystem. This system maintains image and pose temporal histories (time-stamped images and corresponding pose estimates) in a fixed-length first-in last-out queue. This information is available to the remainder of the system. The STAGET control interface permits the STAGET to be repositioned relative to the vehicle and maintains information about the various STAGET parameters and conditions, including information about the current lens aperture and focal length.

All the landmark matching and pose refinement algorithms have been tested extensively, although to a great extent only in indoor domains. A large portion of the original LISP has been ported to C. The plan for the coming year of research is to develop the following behaviors: road following, obstacle avoidance, landmark detection, landmark tracking, and model extension.

Initially, two types of landmark processing behavior will be specified. The first behavior for landmark tracking assumes that a landmark (or set of landmarks) are currently being tracked via the STAGET and all that is necessary is that the vehicle pose be recomputed from the tracked landmarks. However, there are computational tradeoffs as a function of the speed of the vehicle, and the distance and number of landmarks. Thus, not all landmarks may be tracked frame by frame.

The second landmark navigation behavior assumes that no landmarks are currently being tracked and therefore a new landmark must be acquired. This will involve access to a stored 3D model of the campus environment (which initially has been constructed a priori) in order to control the Staget and window on subimages via the Staget. However, the availability and density of landmarks will vary significantly in different areas of the test environment, and therefore model extension will be a necessary goal. Ultimately we seek to demonstrate that an accurate 3D model of the environment can be acquired via exploration in a purely bottom-up manner, while carrying out independent goal-oriented navigation tasks.

1.3. Qualitative Navigation via Image-Based Homing

If the world changes or the robot fails to recognize a landmark, the robot's perception of the world will not correspond to its current map of the world. However, there is ambiguity in whether the errors are in its perception or its map, and if the latter, it must update its map.

Pinette [Pinette 91] has been developing a principled approach to automatic map construction

and maintenance. In place of the usual construction of a geometric map, snapshots of the world at selected target locations along the route are stored as the robot's knowledge of that path. By noting places where a set of memorized routes intersect, a topological "road map" of routes and junctions are represented. To retrace a stored route, a qualitative homing algorithm based on purely local visual servoing is employed to home between successive target locations along the route. This homing algorithm uses no geometric model or positional information; rather, it servos directly on the stored image for a target location, choosing headings that reduce the difference between features of the current bearings and those in the target snapshot. A "consistency-filtering" algorithm has been developed for handling incorrectly matched landmark features [Pinette 92]. It is shown that this algorithm guarantees reliable homing as long as more than two-thirds of the landmarks are correctly identified.

A very robust implementation of a robot navigation system has been developed using image-based homing with a spherical mirror for encoding a 360 degree view at each target location. This navigation system has been implemented as part of an indoor manufacturing automation application domain. It is not yet clear whether these techniques are directly applicable to unconstrained outdoor domains and large-scale space.

2. Motion Analysis

2.1. Multi-Frame Structure from Motion

In robot navigation a model of the environment needs to be reconstructed for various applications, including path planning, obstacle avoidance and determining where the robot is located. Traditionally, the model was acquired using two images (two-frame Structure from Motion) but the acquired models were unreliable and inaccurate. Generally, research has shifted to using several frames (multi-frame Structure from Motion) instead of just two frames. However, almost none of the reported multi-frame algorithms have produced accurate and stable reconstructions for general robot motion. The main reason seems to be that the primary source of error in the reconstruction - the error in the underlying motion - has been mostly ignored. Intuitively, if a reconstruction of the scene is made up of points, this motion error affects each reconstructed point in a systematic way. For example, if the translation of the robot is erroneous in a certain direction, all the reconstructed points would be shifted along the same direction.

Recently, Thomas [Thomas 93a,b] has mathematically isolated the effect of the motion error (as correlations in the structure error) and has shown theoretically that including these correlations in the computation can dramatically improve existing multi-frame Structure from Motion techniques. In several experiments on our indoor robot, the environmental depths of points from 15 to 50 feet away from the camera (and for which ground truth data was available) were reconstructed with errors in the 1-3% range. In one further experiment, the multi-frame full-correlation algorithm was first used to create a model (a set of points) of an indoor hallway from several initial frames of image data. This model was then used to compute the pose of the robot over subsequent frames using Kumar's pose recovery algorithm. The estimated robot pose and actual robot position in the hallway differed by a maximum of three to four inches over a 12.8 foot path.

2.2. Recovering Affine Transforms from Image Sequences

Deformations due to relative motion between an observer and an object may be used to infer 3-D structure. Up to first order these deformations can be written in terms of an affine transform. The recovery of an affine approximation to image deformation has recently been the focus of a large amount of research, and has found application in such disparate areas of computer vision as image stabilization, optical flow computation and segmentation, structure from motion, stereo, and texture, and obstacle avoidance.

Manmatha [Manmatha 93] has developed a technique for measuring the affine transform locally between two image patches using weighted moments of brightness. Unlike previous methods, this technique correctly handles the problem of finding the correspondence between deformed image patches, as is necessary for a correct computation of the affine transform. It is capable of determining affine transforms of arbitrary size, whereas most previous approaches are limited to small transforms. It is first shown that the moments of image patches are related through functions of affine transforms. Finding the weighted moments is equivalent (for the purposes of measuring the affine transform) to filtering the images with gaussians and derivatives of gaussians. In the special case where the affine transform can be written as a scale change and an in-plane rotation, the zeroth and first moment equations are solved for the scale. In experiments on synthetic and real images for this case, the scale was recovered robustly and shown to give reliable

depth estimates. Work is continuing on extending the basic techniques to the general case.

2.3. Multi-Sensor Dextrous Manipulation

Gruppen and Weiss [Gruppen 93] have continued their work on a multi-sensor approach to dextrous manipulation. The goal of this project is the integration of sensing and control for the task of finding a stable grasp configuration for an unknown object. A subgoal is the integration of visual and haptic (proprioceptive) sensory data to incrementally build a model of the object. This approach uses knowledge of the task and the accuracy and completeness of the model to control the sensing actions.

The system consists of a camera mounted on one robot and the Utah/MIT hand mounted on another. The system calibration or identification problem involves computing the transformation from the coordinate system defined by the manipulator robot to the coordinate system defined by the camera robot. The pose determination algorithm of Kumar and Hanson [Kumar 92] has been adapted for this purpose. As the manipulator robot moves, known feature points are tracked. Given the kinematics of this robot, the pose of the camera with respect to the coordinate frame of the manipulator robot are computed and incrementally refined using iterative, extended Kalman filtering. Experiments were performed to demonstrate that the accuracy of the filtering algorithm was comparable to that of smoothing using a least squares fit with all of the data, yet the computation time was much less. An additional feature of the method is that the kinematics of the camera robot can be computed at the same time.

Gruppen and Huber [Huber 92] have obtained 3D surface points from the Utah/MIT hand without the use of tactile sensors. The measurements used are posture, velocities, and torques. This will be integrated with the measurements obtained from the camera sensor.

2.4. Shape Recovery from Occluding Contours

Recovering the shape of an object from two views (e.g. stereo) fails at occluding contours of smooth objects because the extremal contours are view dependent. For three or more views, shape recovery is possible, and several algorithms have recently been developed for this purpose. Szeliski and Weiss [Szeliski 93] have developed a new approach to the multiframe shape recovery problem which does not depend on differential measurements in the image, which may be noise sensitive. Instead, a linear smoother is used to optimally combine all of the measurements available at the contours (and other edges) that are

tracked through the set of images. This allows the extraction of a robust and dense estimate of surface shape and the integration of shape information from both surface markings and occluding contours. The results provide an extremely promising path for recovery of 3D shape models in an industrial setting where the motion is known.

3. Interpretation of Static Scenes

3.1. Learning 3D Recognition Strategies

Most knowledge-directed vision systems are tailored to recognize a fixed set of objects within a known context. Generally, the programmer or knowledge engineer who constructs them begins with an intuitive notion of how each object might be recognized, a notion which is refined by trial-and-error. Unfortunately, human engineering is not cost-effective for many real-world applications. Moreover, there is no way to ensure the validity of hand-crafted systems. Worst of all, when the domain is changed, the systems often have to be rebuilt from scratch.

The Schema Learning System (SLS) [Draper 92, 93b] automates the construction of knowledge-directed recognition strategies. Starting from a knowledge base of visual procedures and object models, SLS learns robust strategies for locating landmarks in images and recovering their positions and orientations, if necessary. Each strategy is specialized to a landmark, taking advantage of its most distinctive characteristics, whether in terms of color, shape, or contextual relations, to quickly focus its attention on the landmark and recover its pose. Furthermore, because SLS learns from experience by a strict generalization algorithm, it is possible to predict both the expected costs and the expected error rates (due to a lemma by Valiant) of the strategies it develops.

3.2. Figural Completion from Principles of Perceptual Organization

Figural completion is the preattentive ability of the human visual system to build complete and topologically valid representations of environmental surfaces from the fragmentary evidence available in cluttered scenes. A description of a grouping system developed by Williams, employing a two-stage process of completion hypothesis and combinatorial optimization, appeared in a previous workshop proceedings [Williams 90]. Preliminary experimental results were also reported. Since that time there has been significant progress in two major areas. First, the mathematical basis for the grouping constraints employed in the optimization stage has been clearly elucidated. This has

allowed a proof of the necessity and sufficiency of the grouping constraints for scenes composed of flat embeddings of orientable surfaces with boundary. Second, a more advanced grouping system which uses cubic Bezier splines of least energy to model the shape of perceptual completions has been implemented. The new system is demonstrated on a number of figures from the visual psychology literature which are beyond the capability of the old system.

3.3. Perceptual Organization of Curvilinear Structure

During the past year, Dolan has continued his work on curvilinear grouping [Dolan 92]. A SIMD implementation of the curvilinear grouping system has been developed, along with a simplified, distributed representation of curves for use in the CAAPP. The integration of multiple grouping processes--in particular, curvilinear and area grouping -- is currently being examined. Many of these ideas are being incorporated in a general grouping module for KBVision, which will facilitate research and experimentation with many diverse forms of grouping.

3.4. Stochastic Projective Geometry

The use of projective invariants for object recognition and scene reconstruction has been the subject of intense interest in the image understanding community over the past few years. Although classic projective geometry was developed with mathematically precise objects in mind, practical applications must deal with errorful measurements extracted from real image sensors. A more robust form of projective geometry is needed, one that allows for possible imprecision in its geometric primitives. In his Ph.D. thesis [Collins 93], Collins represents and manipulates uncertain geometric objects using probability distributions in projective space, allowing valid geometric constructions to be carried out via statistical inference. The result is a methodology for scene reconstruction based on the principles of projective geometry, yet also dealing with uncertainty at a basic level. The effectiveness of this framework has been demonstrated on several geometric problems, including the derivation of 3D line and plane orientations from a single image using vanishing point analysis, the extraction of a planar patch scene model using stereo line correspondences, and the reconstruction of planar surface structure using multiple images taken from unknown viewpoints by uncalibrated cameras.

More specifically, Collins shows that projective N -space can be visualized as the surface of a unit sphere in $(N+1)$ -dimensional Euclidean space.

Each point in projective space is represented as a pair of opposing or *antipodal* points on the sphere. By the identification of projective space with the unit sphere, antipodally symmetric probability distributions on the sphere may be interpreted as probability distributions over the points of projective space, and standard constructions of projective geometry can then be augmented by statistical inferences on the sphere. Probability densities defined in this way can also be used for representing uncertainty in unit vectors, orientations, and the space of 3D rotations (via unit quaternions).

3.5. Shape from Shading

Oliensis' previous work on shape from shading [Oliensis 92] has been extended in a number of ways. First, while our earlier work usually assumed that the illumination was from the direction of the camera, the shape reconstruction algorithms and convergence proofs have been extended more recently to the case of illumination from any direction [Oliensis 93a]. As before, these algorithms are provably and monotonically convergent, and (in many cases) can be shown to converge to the correct surface. Moreover, it has been shown that a whole family of algorithms could be developed, and that all would give equivalent surface reconstructions. This is convenient since some of the algorithms are better for theoretical analysis while others are more efficient in practice. The uniqueness proofs for the surface given the shaded image, and the corollary that regularization is not necessary for shape from shading, have also been extended. Experimentation with these algorithms on synthetic and real images show that they are fast and robust, taking less than 10 seconds on a DECstation 5000 for a 200 x 200 real image.

These algorithms still require that a small amount of information on the surface be provided, namely: 1) a list of those singular points (the brightest image points) corresponding to local minima of the surface height (as opposed to the other possibilities of a local maximum or a saddle point); and 2) the heights of these singular points. However, in a second extension of previous work [Oliensis 93b], Oliensis has developed a new algorithm that is capable of determining this information automatically, and thus can reconstruct a general surface from shading with no a priori information on the surface. In experimental tests on complex synthetic images, this algorithm has produced good surface reconstructions over most of the image. For 128 x 128 images, the reconstruction takes less than 30 seconds on a DECstation 5000. Moreover, the algorithm appears noise resistant, giving good

reconstructions even in the extreme case of an added pixel noise of 10%. It appears that it will also be possible to prove the convergence of this algorithm to the correct surface in the limit of perfect resolution.

All algorithms thus far have assumed that the imaged surface was matte. Even with this restriction, the algorithms are potentially useful in controlled industrial or research applications. At UMass these algorithms will be ported to the robotics laboratory environment, and used in combination with other means of shape sensing and recovery to aid in research in grasping partially or unmodeled objects. Further extensions include adapting the current algorithms to the realistic case of a partially specular surface. With this extension, shape from shading could become practical for a variety of applications.

4. Image Understanding Architecture (IUA) Overview

Work on the IUA [Weems, 1993] has advanced in three areas in the preceding year: compilers and system software, hardware and architecture, and applications and algorithms. The IUA is a tightly coupled, heterogeneous parallel processor being developed by UMass, Hughes Research Labs, and Amerinex Artificial Intelligence (AAI) under DARPA funding. It is intended to support real-time knowledge-based vision applications and research by providing three distinct parallel processors in a single architecture: a fine-grained SIMD/Multi-associative array for low-level vision, a medium-grained SPMD array for intermediate-level symbolic vision, and a coarse-grained multiprocessor for high-level, knowledge-based processing. A proof of concept prototype of the IUA was constructed under a previous effort and the current work is directed at developing a second generation of the system with enhanced performance and the ability to be fielded in the DARPA Unmanned Ground Vehicles (UGV) program.

4.1. IUA Compilers and System Software

AAI has completed development of the C++ class library for the low level of the IUA. The class library defines a set of image plane types upon which parallel operations may be performed. Work at AAI includes the incorporation of additional optimization code into the Gnu C++ compiler so that image planes are treated more like first-class objects in C++. An automated test system has also been developed for the machine's microcode library, to facilitate regression testing of new releases. For the intermediate-level processor, basic operating system support, multitasking, and messaging have been

implemented on a TMS320C30 Single Board Computer (SBC), and recently these were transported to another SBC with two TMS320C40 processors that are configured to simulate the intermediate level of the IUA. A debugger has also been implemented for the intermediate level. Work is now under way to transport the KBVision™ system to the IUA.

UMass has implemented a version of the Apply language for the low-level processor of the second generation IUA. The compiler generates code compatible with the C++ class library. It permits us to easily import image processing operations written for the CMU Warp or Intel iWarp machines.

4.2. IUA Hardware Status

The prototype IUA has been running at Hughes for most of the last year. Under the prototype development contract, only a very simple controller was built to demonstrate the basic functionality of the processor arrays. It was never intended that the prototype controller be fully programmable. However, Hughes and Amerinex AI invested additional effort to develop software that allows C++ code for the second generation to execute on the prototype hardware. Because of the nature of the controller, instructions can only be issued at VME bus rates to the array, which is significantly slower than the array can accept them. However, it does permit demonstration of the functionality of the array hardware on real applications. Hughes has since implemented the low-level portion of the DARPA IU Benchmark, an SDI application, and an ATR application on the prototype.

The custom chips used in the IUA have been fabricated and are undergoing testing. Each chip contains 256 bit-serial processors with on-chip cache, and has roughly 600,000 transistors. The system's array control unit, backplane, and chassis have been built and tested. Processor and memory boards are currently under construction. The I/O subsystem for the machine has also been designed, and will support the equivalent of 20 simultaneous sensor inputs at 512 X 512 X 8-bit resolution with automatic mapping onto the processor virtualization scheme used for the low level, with almost no latency. The I/O subsystem will also support the selection of multiple regions of interest from an image. Hughes has indicated that the first machine should be assembled by the end of February, 1993.

Work has already begun on the analysis and design for the third generation IUA. UMass has developed a system for capturing traces of programs written in the C++ class library as they

execute on an abstract parallel machine. The traces are then fed to a simulation system that models hardware architectures with different features and parameters. The system allows us to gather real performance data for different architectural configurations, and to analyze the data statistically. The performance data will then be contrasted with cost estimates for the different configurations to produce a specification for the third generation IUA.

4.3. IUA Applications and Algorithms

The low-level processor of the IUA is a square mesh of processing elements, augmented with a second (reconfigurable) mesh, called the Coterie Network. This network allows the mesh to be partitioned, for example, into areas corresponding to regions in an image. One particularly useful operation is the ability to enumerate elements within a partition or to summarize (reduce) the information in a partition to a single value. The parallel prefix operation is the general form of this type of operation. It is especially desirable to be able to carry out parallel prefix in all partitions at once, i.e. to perform a multi-prefix operation [Herbordt, 1992]. An algorithm has been developed for multi-prefix that is significantly faster than alternatives using general purpose routing in the mesh.

As recommended by the DARPA IU Benchmark Workshop participants, much of the benchmark [Weems, 1988, 1990] has been recoded as a set of library routines which are called by the core of the benchmark. We have also begun developing the second level benchmark, which will incorporate tracking of moving objects over a sequence of images. The goal of the new benchmark is to test system performance over a longer period of time so that, for example, caches and page tables will be filled. The benchmark will also explore I/O and real-time capabilities of the systems under test, and involve more high-level processing.

UMass has developed a parallel algorithm for the IUA that computes a dense depth map for a scene from a pair of images taken by a moving sensor [Dutta 93]. The algorithm has an average error of about 8 percent in depth, as computed from randomly sampling points corresponding to objects in the scene with known distances from 21 to 76 feet from the camera. The experiments were done with fairly large displacements (four feet of forward motion between the images) so that a large (41 X 41 pixel) search window was required to establish correspondences, resulting in 1681 image-to-image correlations being performed. In simulations of the second generation IUA, it was determined that the execution time will be about

0.54 seconds, of which 0.53 seconds is taken up solely by the correlations. We are thus looking into approaches in which an estimate of the motion is available or in which a series of frames with smaller displacements can be used (allowing the search window to be constrained).

4.4. Line Extraction

UMass has also developed a parallel algorithm for extracting straight lines from an image. Using the second generation IUA simulator, the algorithm executes in as little as 31 milliseconds for images that map to the array with a 1:1 virtualization ratio. We are currently evaluating the quality of the results, but a preliminary examination indicates that the algorithm gives very consistent lines over sequences of images, which is an important attribute in the support of algorithms that use line tracking.

5. Image Exploitation under RADIUS

UMass is developing mechanisms for site model acquisition, extension and refinement [Collins 93a] based on technology that has already proven effective in the mobile robotics domain.. Automatically acquiring the initial 3D site models from scratch is a challenging problem that will be the focus of future research. Our current work assumes that a partial model of the site is provided apriori by the image analyst. Our model-based refinement and extension algorithms are then applied to automatically correct inaccuracies in the initial site models, and extend them to include previously unmodeled cultural features (buildings, roads, etc.) based on information from new images.

Rather than building a turn-key system, UMass will be delivering a set of modules for performing specific tasks of direct benefit to the image analyst. The following is a list of the early deliverable modules that are currently being evaluated on the model board test imagery supplied to the research community.

1. Feature Extraction Module. This module condenses the vast amount of information in each image into a manageable set of symbolic descriptions. Two straight line extraction algorithms are being evaluated: the Burns algorithm based on fitting planar patches to the underlying image intensity surface, and the Boldt algorithm for hierarchical geometric edge grouping. Also under development are routines for extracting curved lines, and for locating dihedral and trihedral junctions to subpixel accuracy.

2. Model Matching Module. Given the approximate pose (location and orientation) of the sensor, a partial 3D wireframe site model, and a set of extracted straight lines, the best match of the projected 3D model to the line data will be found using a novel model matching algorithm due to Beveridge.

3. Model Extension Module. Given a partial model and a set of model-data feature correspondences over multiple images, the site model will be extended to include further unmodeled features. Two techniques are being evaluated. The first is based on recovering the camera pose using a robust pose estimation technique due to Kumar. This algorithm is effective even when significant numbers of feature correspondences are in error. Using the computed pose for multiple images from multiple viewpoints, the 3D positions of unmodelled features are found by triangulation. A second approach is based on direct estimation of the 3D to 2D projective transformation relating model features to image features. The benefits of this approach are that multiframe triangulation can still be performed without first solving for camera pose, and without relying on accurate knowledge of the internal camera parameters.

4. Vanishing Point Module. Vanishing point analysis is a flexible tool for geometric reasoning in cultural domains. Among its many uses are the determination of 3D line and plane orientations, refinement of extracted linear features based on convergence constraints, pose estimation, and camera calibration. An efficient vanishing point detection and estimation algorithm due to Collins and Weiss is being evaluated.

In addition to developing new techniques for automatically acquiring initial site models, new research will investigate statistical techniques for applying projective invariants to the modeling process to accurately derive structure without explicit camera models or knowledge of viewpoint. Initial experiments in this direction have yielded promising results. Other encouraging results have been obtained regarding the difficult problem of image to image registration. A technique based on vanishing point analysis [Collins93b] allows an oblique aerial view to be rectified (unwarped) to present a simulated vertical view, allowing full perspective aerial images to be registered with a computationally tractable affine matching approach.

6. References

J. R. Beveridge and E. M. Riseman. Hybrid weak-perspective and full-perspective matching, Proc. CVPR, 1992.

- R. Collins, Allen R. Hanson, Edward M. Riseman, Yong-Qing Cheng, Model Matching and Extension for Automated 3D Site Modeling, Proc. 1993 DARPA IUW, Washington, DC. April 19-21, 1993a.
- R. Collins and J. Ross Beveridge, Matching Perspective Views of Coplanar Structures using Projective Unwarping and Similarity Matching, Proc. 1993 DARPA IUW, Washington, DC. April 19-21, 1993b.
- R. Collins, forthcoming Ph.D. thesis, 1993.
- J. Dolan and E. Riseman, Computing Curvilinear Structure by Token-Based Grouping, IEEE CVPR, Champaign, IL, June 1992, pp.264-270
- B. Draper and A. Hanson, ISR3: A Token Database for Integration of Visual Modules, Proc. 1993 DARPA IUW, Washington, DC. April 19-21, 1993a.
- B. Draper and E. Riseman, The Schema Learning System, Proc. 1993 DARPA IUW, Washington, DC. April 19-21, 1993b.
- B. Draper, A. Hanson, and E. Riseman, Learning Blackboard-Based Scheduling Algorithms for Computer Vision, Computer Science Department, University of Massachusetts, Technical Report TR92-51, August 1992.
- R. Dutta, C. Weems, and E. Riseman, Parallel Dense Depth from Motion on the Image Understanding Architecture, Proc. 1993 DARPA IUW, Washington, DC. April 19-21, 1993.
- R. Grupen and R. Weiss, "Sensor-Based Incremental Planning for Multifingered Manipulators, Grantees Conference of the NSF Division of Design and Manufacturing, Univ. of North Carolina, January 1993.
- M. Herbordt and C. Weems, Computing Reduction and Parallel Prefix Using Coterie Structures., Proc. IEEE Conf. on the Frontiers of Massively Parallel Processing, October, 1992, pp. 141-149.
- Huber, M., and Grupen, R., "2-D Contact Detection and Localization Using Proprioceptive Information", Computer Science Technical Report 92-59, University of Massachusetts, Amherst, August, 1992.
- Kumar, R. Model Dependent Inference of 3D Information from a Sequence of 2D Images, Ph. D. Thesis and Technical Report TR92-04, Department of Computer Science, University of Massachusetts (Amherst), 1992.
- R. Manmatha and J. Oliensis, Measuring Affine Transforms - I: Recovering Scale and Rotation, Proc. 1993 DARPA IUW, Washington, DC. April 19-21, 1993.
- J. Oliensis and P. Dupuis, Provably Convergent Algorithms for Shape from Shading, Proc. DARPA IUW, Washington, DC. April 19-21, 1993a.
- J. Oliensis, A General Algorithm for Shape from Shading, International Conference on Computer Vision, Berlin, Germany, May 11-13, 1993b, to appear.
- J. Oliensis and P. Dupuis, Direct Method for Reconstructing Shape from Shading, Computer Science Department, University of Massachusetts, Technical Report TR92-32, May 1992
- Brian Pinette. Qualitative navigation. II: Handling inconsistently-matched landmarks. Proc. SPIE Volume 1831, Mobile Robots VII, Boston, MA. 1992.
- Brian Pinette. Qualitative homing. Proceedings of IEEE Symposium on Intelligent Control, Arlington, VA. 1991.
- Sawhney, H., "Spatial and Temporal Grouping in the Interpretation of Image Motion", Ph.D. Thesis and Technical Report 92-05, Computer Science Department, University of Massachusetts, Amherst, February, 1992.
- H. Sawhney and A. R. Hanson. Affine trackability aids obstacle detection., Proc. CVPR, Champaign, IL, June 1992, pp. 418-424.
- H. Sawhney, R. Kumar, A. Hanson, E. Riseman, Landmark-Based Navigation - Model Extension and Refinement, Proc. DARPA IUW, Washington, DC, April 19-21, 1993.
- R. Szeliski and R. Weiss, Robust Shape Recovery from Occluding Contours Using a Linear Smoother, Proc. 1993 DARPA IUW, Washington, DC. April 19-21, 1993.
- J. Thomas, A. Hanson, and J. Oliensis, Understanding Noise : The Critical Role of Motion Error in Scene Reconstruction, Proc. ICCV, Berlin, Germany, May 11-13, 1993a, to appear; (A similar paper appears in these proceedings).
- J. Thomas, "Obtaining the Robot Path Using Automatically Acquired Models, Proc. International Conference on Intelligent Autonomous Systems (IAS), Pittsburgh, PA, February 1993b, to appear.
- C. Weems, M. Herbordt, R. Dutta, K. Daumuceller, G. Weaver, S. Dropsho, J. Burrill, R. Lerner, A. Hough, Progress and Current Research in the Image Understanding Architecture, Proc. 1993 DARPA IUW, Washington, DC. April 19-21, 1993.
- Weems, C.C., Riseman, E.M., Hanson, A.R., Rosenfeld, A., An Integrated Image Understanding Benchmark for Parallel Processors (Invited Paper), Journal of Parallel and Distributed Computing, Vol. 11, pp. 1-24, 1990.
- Weems, C.C., Hanson, A.R., Riseman, E.M., Rosenfeld, A., An Integrated Image Understanding Benchmark: Recognition of a 2-1/2D "Mobile", Proc. CVPR, Ann Arbor, MI, June 5-9, 1988.
- Williams, L. R. (1990). "Perceptual Organization of Occluding Contours," Proc. of DARPA IUW, Pittsburgh, PA, pp. 639-649.

Progress in Image Understanding at MIT

W. E. L. Grimson, B. K. P. Horn, T. Poggio and staff
MIT AI Lab
Cambridge MA 02139

ABSTRACT

Our program in Image Understanding has maintained a primary focus on issues in object recognition, especially the problems of selection, indexing, saliency computation and integration of visual cues, and a secondary focus on navigation. We have also continued our work on the computation and the use of low level visual cues such as motion, stereo, color and texture, on analog VLSI circuits and on learning.

1 Introduction

Image Understanding research at the MIT AI Lab has continued along a range of fronts, from low level processing, such as stereo, motion, color and texture analysis, through intermediate stages of integration of visual information, to higher level tasks such as object recognition and navigation. This report summarizes our main recent accomplishments in these areas. As is usual in these reports, we refer interested readers to other publications for more details.

2 Object Recognition

Because it has been one of our central focal points, we begin with our recent work in object recognition. In approaching the problem of recognizing objects from noisy images of cluttered scenes, we have found it convenient to separate out several different aspects of the problem:

- **Selection:** Given a large set of image features from a cluttered scene, select (or group) subsets likely to have come from single objects, and use a rank ordering to place the most salient ones first.
- **Indexing:** Given one of these image feature subsets, select a small set of object models from the library that are likely to match the data.
- **Matching:** Given a data feature subset and an object model, determine if there is a legal transformation that would carry the model into a pose in the image that is consistent with the data, possibly by finding a matching between data and model features. It is often useful to separate this stage into two subproblems:
 - **Hypothesize** possible solutions, using minimal model and image information.
 - **Verify** such hypotheses, using more detailed information.

We will describe our recent work in each of these areas.

2.1 Selection and Attention

We have argued for some time that robust and efficient solutions to the selection (or grouping) problem are essential to practical recognition systems. Earlier work, using both formal analysis and experimental studies [22; 13; 27], has shown that the complexity of many approaches to recognition are dramatically reduced if decent selection is provided, and that the false positive/false negative rates for such methods are also significantly improved with good selection.

One advantage of focusing on the issue of selection for recognition is that it provides constraints on the requirements of early processing stages. For example, cues such as color or texture are often considered from the viewpoint of extracting object surface measurements, which requires that one account for illumination and other scene effects in inverting the image measurements to obtain object parameters. If one simply wants to use these cues to separate regions of an image likely to have come from a single object, much less stringent requirements are placed on the task, leading to simpler and hopefully more robust algorithms.

Towards this end, Tanveer Syeda-Mahmood has recently completed a Ph.D. thesis [46] that explores the role of cues such as color and texture in selection for recognition. She does this by developing and implementing a computational model of visual attention, which serves as a general purpose selection mechanism in a recognition system.

The approach supports two modes of attentional behavior, namely *attracted attention* and *pay-attention* modes. The attracted attention mode of behavior is spontaneous and is commonly exhibited by an unbiased observer (i.e., with no a priori intentions) when some object or some aspect of the scene attracts his/her attention, while the latter is a more deliberate behavior exhibited by an observer looking at a scene with *a priori* goals (such as the task of recognizing an object, say) and hence paying attention to only those objects/aspects of a scene that are relevant to the goal.

Briefly, the model suggests that the scene represented by the image be processed by a set of interacting feature

detectors that generate a hierarchy of maps, representing features such as brightness, color, texture, depth, grouping of edges, and others such as shape, size, symmetry, etc. The feature maps are then processed by filters incorporating strategies for selecting distinctive regions in these maps. The choice of these strategies is guided by a central control mechanism that combines top-down task level and *a priori* information with the bottom-up information derived from the features, to demonstrate either mode of attentional behavior as desired. Finally, an arbiter module housing another set of strategies selects the most significant features across the feature maps, which can then be used in an object recognition system.

A system implementing the computational model described here was built using three features: color, texture, and parallel-line-groups. The respective feature maps were built, and the selection filters for finding distinctive regions in these maps have been developed. In addition, a version of the arbiter module to combine the saliency information from the various features has been built.

Because we are interested mainly in separating regions likely to have come from a single object, we do not need to exactly recover object parameters such as body color. Rather, we can focus on methods that describe the color image as consisting of perceptually different colored regions. This can be done by focusing on the components of a color signal that are most relevant to human categorization of colors (e.g. saturation and brightness, but not hue). Syeda has developed such a method of perceptual categorization of a color-space, which supports fast color region segmentation. A color saliency map was then built which used a color saliency measure that emphasized attributes that are also salient in human color perception. The key point is that such a saliency measure serves to highlight regions of interest for a recognition system.

The texture feature map was generated by regarding the image as being generated by a space-limited stationary stochastic process. Here, the segmentation of the textured image was obtained by a comparison of the linear prediction spectra of adjacent windowed regions of the image. Properties such as the relative distribution of dark and bright blobs were then made use of to judge the distinctiveness of a region. This was used to generate the texture saliency map.

Lastly, the parallel-line-groups feature map highlighted groups of closely-spaced parallel lines in an edge image. It has been found that some texture information can be modeled this way. For example, printed letters on a surface (such as a bottle) appear as a bunch of closely spaced parallel lines when passed through an edge detector. Similarly, some types of wooden tables show this type of texture in an image.

These feature maps can then be combined to isolate regions of an image for analysis by a recognition system (in our implementation this was a combination of a tree search algorithm and a linear combinations approach). The feature maps and their associated saliency maps can be driven in a pay-attention mode, in which the color and texture information in the model (extracted using

the feature maps described earlier) is used to build a description of the object-model. This description was then used to design strategies for the selection filters. This involved developing new algorithms for finding instances of regions in the image satisfying object-model color and texture descriptions. Such regions are then passed to the recognition system for analysis. Experimental results show that the methods drastically reduce the complexity of the recognition process by rejecting clutter from consideration. The system can also be driven in attract attention mode, in which the most salient portions of the scene are analyzed first, again reducing the complexity of the recognition stage. An example is shown in Figure 1.

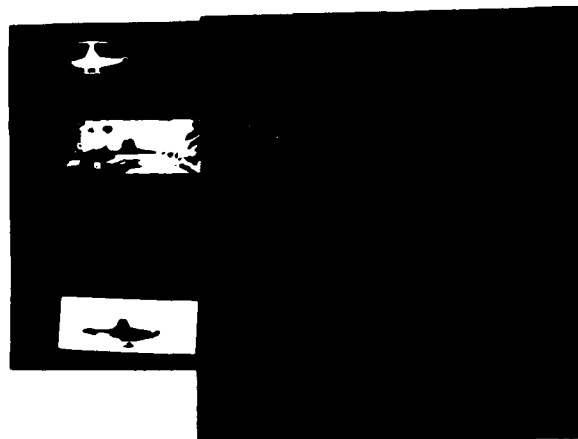


Figure 1: Example of attentional selection for recognition. Top Left: Image used to create the model. Middle left: Scene. Bottom Left: Selected salient color regions. Top center: Model showing corner and line features. Middle center: Edge image of scene, showing features. Bottom center: saliency features. Top right and middle right: matched features found by algorithm. Bottom right: alignment of model with image.

The key results here are a framework for combining sensory information to support recognition, the use of an attention mechanism to select targets for recognition, and novel methods for handling texture and color information.

2.2 Indexing

Even if we can isolate key regions of an image, we still need to know what objects may be present. In some tasks, we are looking only for a specific target, or a small number of targets, in which case model-driven selection can be used to isolate regions of interest. In other cases, we need to consider large libraries of objects, in which case we need some means of using selected image features to identify subsets of the library as candidate models. Cues such as color and texture, discussed above, can help with this object indexing. In general, however, other information is also needed.

David Jacobs, as part of his recently completed Ph.D. thesis [27], has shown that simple aspects of an object's shape can often be used to efficiently index objects in a library. Jacobs' method views the indexing problem as stating: can one compactly represent the space of all images that an object model can create, given the type of projection model used? If one can, then to handle the indexing problem, we can precompute each model's manifold of possible images in an image space. At recognition time, we can compute the associated representation of the current image, use this to access image space, and retrieve all models that could have caused it. The key question is whether one can actually represent all possible images of a model in an efficient way. Jacobs has shown, perhaps surprisingly, that in several nontrivial cases, one can.

The results are summarized as follows. We assume that a 3D object is transformed by an arbitrary affine transformation, followed by a scaled orthographic projection. For the case of 3D points, this is equivalent to applying an arbitrary 3×3 matrix to the points, then translating them, then projecting them. To describe the images that a model can produce under this class of transformations, we first define image space, then determine the shapes of the model manifolds. Jacobs argues for using affine coordinates to represent image space. In particular, if one selects three ordered point features to establish a basis, then all other points can be written in terms of coordinates with respect to this basis: that is if q_1, q_2, \dots, q_n denote the image points, and if

$$o = q_1 \quad u = q_2 - q_1 \quad v = q_3 - q_1$$

are the affine basis, then all other points can be represented by coordinates (α_i, β_i) by

$$q_i = o + \alpha_i u + \beta_i v.$$

These coordinates are invariant to any affine transformation, and hence an image is uniquely identified by

$$o, u, v, (\alpha_4, \beta_4), (\alpha_5, \beta_5), \dots$$

It turns out that the first three vectors do not provide any information about whether a scene could produce this image, so we use only the (α_i, β_i) parameters to represent an image. Thus, an image with n ordered points maps to a point in a $2(n-3)$ dimensional space, which can be divided into two orthogonal $n-3$ dimensional spaces, one for the α coordinates and one for the β coordinates. The advantage of doing this, as Jacobs has shown, is that the set of all images that a model of n ordered points could produce is simply a pair of parallel lines, one in each space. In this case, indexing simply says, given an image basis, compute the affine coordinates of the points, then find all model lines that pass through the associated point in $\alpha-\beta$ space. One must allow for uncertainty in the measurements, but this can be shown to be easily handled and simply expands the image points to small regions in the α and β spaces [21]. An example of the indexing system correctly retrieving candidate models is shown in Figure 2.

Extensions of this approach to deal with other types of features are discussed in an article by Jacobs in these

proceedings. We note as an aside that considering this model of linear transformations has proved fruitful in other problems, such as the linear combinations approach to recognition [48] and in dealing with affine structure from motion [28; 41].

The key result here is a very efficient method for handling indexing for some classes of objects, as well as a novel framework for investigating the interactions between object structure and image projections.

2.3 Matching

A central part of recognition, once we have found subsets of image features of interest and sets of models of interest, is to determine whether the image features are in fact consistent interpretations of the model features. Over the past several years, we have developed a variety of different approaches to this problem, including Interpretation Tree Search [22], Alignment [24; 25] and Linear Combinations [48]. Here we report on some new alternatives to these approaches, as well as improvements on these approaches.

2.3.1 Making Alignment Robust

The alignment approach to recognition [24; 25; 6] proceeds by matching a small set (typically 3) of image features to model features, using this match to determine the associated transformation of the object (modeled as a weak perspective transformation), and projecting the remaining model features into the image for verification. In the original system, uncertainty in the image measurements was dealt with in a somewhat ad hoc manner. Recently [21] we have shown how to analyze the effects of that uncertainty on the set of possible transformations. Alter [1] has extended that work to supplement alignment approaches with a verification stage that is guaranteed to be correct. In particular, he shows that using a bounded error model on the image features, one can compute the range of image positions for all other model features, both for planar and solid objects, and for point and line features. This allows one to exactly specify the range of image positions over which to search for matching features, so that one will not miss any supporting evidence, while at the same time keeping the chances of false matches minimal. One can further extend this approach by adding a Bayesian analysis of the actual matching regions, so that one can determine the likelihood of each verified match actually being correct. This allows one to determine the best regions in which to search for features, by determining those most likely to contribute to a correct interpretation. An example of the image search regions is shown in Figure 3. Extensions of the method to line features has also been done, and results show, as expected, that lines are considerably more powerful as verification features than points.

A related result concerns the models of sensor uncertainty used both in the analysis of recognition methods and in the derivation of verification and likelihood techniques. Most of our earlier work has been based on a bounded error of sensor uncertainty. Karen Sarachik has been working on the problem of estimating the effects of sensor noise on the problem of model based object recognition, for other classes of uncertainty. For the analysis.

it is assumed that the location of a point feature in an image is corrupted by noise, which is modeled as a two dimensional Gaussian distribution, and the presence of the model in the image is posed as a binary decision problem. The positional uncertainties of the point features are traced through the recognition algorithm, resulting in analytic expressions for the confidence level of the algorithm's decision. Until now the analysis has been completed only for one algorithm, geometric hashing as introduced by Wolfson, Lamdan, Schwartz and Hummel. Using this technique it is possible to explicitly compare the expected performance of different recognition algorithms and noise models, a useful tool for the domain of multi-sensor fusion.

2.3.2 Pose Space Analysis

In earlier proceedings, we have reported on our work in developing recognition algorithms that work directly in the space of possible poses of an object, rather than in the space of feature correspondences. In the ideal case of perfect sensor data, one can simply search over all possible pairings of model and image features, compute the associated transformation and vote for that transformation in pose space, a la Hough transforms. When uncertainty is allowed in the measurements, however, one must be more careful about voting for the entire volume of transformations consistent with the pairing of a noisy sensor measurement and a model feature, and this increases the demand on searching fine tessellations of the pose space.

Todd Cass has recently complete a Ph.D. thesis that presents an elegant way around this problem, by exploiting the geometry of pose space directly. Cass [10] has provided a formulation of the problem in which one can develop a polynomial-time algorithm that guarantees finding all feasible interpretations of the data, modulo uncertainty, in terms of the model. The approach is based on representing the model and the sensory data in terms of local geometric features such as vertices and line segments. He assumes bounds on the uncertainty in the position or orientation of the data features due to sensor error. He then shows that there are only a polynomial number of quantitatively different transformations that align the model and the data modulo error. Object localization is accomplished using a polynomial-time search through the set of all model transformations to find those that align large subsets of model and data features within the uncertainty bounds.

Intuitively, this approach can be considered as follows. For each pairing of a data and model feature, there is a set of transformations that will align the model features within the uncertainty region about the data feature. This set of transformations carves out a volume in pose space. If we consider all pairings of data and model features, we get a set of such volumes, and we are interested in finding points in the pose space contained within the intersection of a large number of such volumes. One could find such points by simply sampling points in pose space at some fine spacing, a method used earlier by Cass in implementing a very fast recognition scheme on the Connection Machine. It turns out, how-

ever, that one can efficiently find such volumes by decoupling the search over the full pose space into a coupled search over the translational components and a second search over the rotational components. Moreover, one can use the structure of these geometric arrangements to find very efficient, polynomial-time algorithms for finding the boundaries of these pose-space volumes. Cass has extended his earlier work to allow for unknown scale factors, unknown uncertainty values, and has used results from computational geometry to provide efficient algorithms for exploring pose space.

2.3.3 RAST algorithm

An alternative to Cass' approach to analyzing Pose Spaces is the RAST algorithm (Recognition by Adaptive Subdivision of Transformation Space: [8]). The RAST algorithm solves bounded error recognition problems efficiently.

Bounded error recognition is one of the most commonly used formulations of the visual object recognition problem and has proven its utility in a number of practical systems (for further references and related results, see, for example, [4], [22]). The simplest form of the bounded error recognition problem is the following: given a set of model features (points in R^2 or R^3) and a set of image features (points in R^2), find maximal subsets of image and model features that can be brought into correspondence under given error bounds using rigid body transformations.

The recognition algorithm is based on the idea of carrying out matching with variable sized error bounds: if, for a given set of transformations, a good match cannot be found for large error bounds, then matches with smaller error bounds need not be examined. The RAST algorithm uses particularly convenient representations for sets of transformations that make it simple to implement, efficient, and flexible in the kinds of features and similarity measures that can be used with it.

So far, we have applied the RAST algorithm to 2D recognition problems that involve a very large number (thousands) of very simple image features (short line segments). In such applications, the RAST algorithm is found to be faster than alternative methods (recognition by alignment, Hough transform, search, or correlation). Actual applications using the RAST algorithm include the prototype of a view-based 3D object recognition system and a system for handwritten optical character recognition (OCR). Examples are shown in Figure 4.

2.3.4 View-Based Representations

For curved objects, both the visibility and the location of object parts/features in the image varies in a complicated way with the viewpoint. This has made the development of efficient 3D object recognition algorithms difficult.

In order to avoid these difficulties, many 3D object recognition systems have heuristically used *view-based representations*, i.e., representations that encode object properties and shape from a large number of different viewpoints.

However, using view-based representations only solves the 3D object recognition problem approximately. In

order to understand the nature and significance of this approximation, we have formalized the notion of view-based representations and established error and complexity bounds on the performance of recognition systems that are based on view-based representations[9].

The theoretical results suggest that, for the purposes of object recognition from 2D images, view-based representations are good approximations to true 3D shape representation. Furthermore, we have established model-independent upper bounds on the number of views needed in order to represent a model in a view-based system. Finally, a complexity analysis suggests that view-based recognition can be carried out more efficiently than 3D shape-based recognition.

These theoretical results are supported by a number of simulations and experiments on real images.

2.3.5 Statistical Recognition

An alternative approach to recognition is to treat it as a problem of optimal estimation. Sandy Wells has recently completed a Ph.D. thesis [53] that develops and tests a framework for statistical object recognition.

To formalize this, let the image that is to be analyzed be represented by a set of v -dimensional point features

$$\theta = \{\theta_1 \theta_2 \dots \theta_n\} \quad , \quad \theta_i \in \mathbb{R}^v$$

The model to be matched is also described by a set of point features, these are represented by real matrices:

$$M = \{M_1 M_2 \dots M_m\}$$

To solve the recognition problem we need to find a legal match between image features and model features. Here, legal means that there is some physically meaningful way of positioning the model in the scene so that it would produce image features similar to those actually observed. We can treat this as an optimization problem, wherein we seek to estimate two sets of parameters: the correspondences between image and model features, and the pose of the model instance in the image. The correspondences are described by an interpretation vector

$$\Gamma = [\Gamma_1 \Gamma_2 \dots \Gamma_n] \quad , \quad \Gamma_i \in M \cup \{\perp\}$$

Here $\Gamma_i = M_j$ means that image feature i corresponds to model feature j , and $\Gamma_i = \perp$ means that image feature i is due to the background.

The pose of the model instance in the image, β , is a real vector. An associated projection function P is defined:

$$P(M_i, \beta) \in \mathbb{R}^v$$

P maps model features into the v -dimensional image according to the model's pose.

Our goal is to obtain estimates of the correspondences and pose by maximizing the posterior density with respect to Γ and β , as follows

$$\widehat{\Gamma, \beta} = \arg \max_{\Gamma, \beta} p(\Gamma, \beta | \theta)$$

In other words, we want to find the assignment of model features to image features, and the related pose (position and orientation) of the object that optimally accounts for the observed data. We can treat this as an

estimation problem, and use Baye's rule to calculate the a-posteriori probability density of Γ and β :

$$p(\Gamma, \beta | \theta) = \frac{p(\theta | \Gamma, \beta)p(\Gamma, \beta)}{C}$$

where C is a normalizing constant independent of Γ and β . Then we seek estimates for Γ and β that optimize this objective function.

Note that we couple the effects of the objects pose directly into the matching problem. There are, of course, some sensor features that are not directly pose related, such as the fractal dimension of a region. These features can also be incorporated into the matching process, either as priors on the correspondence, or as filters that remove some correspondences directly.

To solve this optimization problem, we need several things. First, we need to model θ . This can be done by a careful physical modelling of the sensor, by taking into considering the effects of noise on the transduction process, and providing careful models of the distribution of uncertainty about the measured values. Such models can be derived for widely varying sensors, other than visual, and in we are in the process of applying this approach to LADAR and SAR sensors.

As an example, one simple model is to assume that the probability density function on features is uniform for features arising from the background, and is normally distributed about their predicted locations in the image for matched features. Of course, this is a simple model. For some types of features, we have more explicit models of the distribution of the feature, which will simply replace the variance of the normal distribution.

Second, we need to model the probability of an interpretation (or matching of features) and the probability of a pose. One simple method is the following. The probability that a feature belongs to the background is B ; the remaining probability is uniformly distributed for correspondences to the m model features.

Our simple model also assumes that prior information on the pose is a normal density. With this choice for the form of the pose prior, the system is closed in the sense that the resulting pose estimate will also be normal. This is convenient for coarse-to-fine techniques (or multi-resolution methods), in which we use coarse data to get an initial estimate, then refine this by focusing on subportions of finer resolution data.

If little is known about the pose a-priori, the prior may be made quite broad. This is expected to often be the case. Note, however, that better models would incorporate additional information about the scene. For example, if we know the parameters of the ground plane, and the target is known to be in a stable position on that plane, we should be able to incorporate this knowledge into better priors on the pose parameters. For example, if range information is also known, then only two of the six parameters of pose are completely unknown. The others can be constrained from such additional information, thereby reducing the complexity of the search for a match.

If we assume independence of the correspondences and the pose (before the image is seen), the composite prior

is a straightforward product of the prior distribution on pose and the probability distributions on matched and unmatched features. Thus, in our simple example, we can describe the probability of a pose and a correspondence in terms of measurable quantities in the data.

Given this, we need efficient methods for finding optimal estimates for the parameters of interest. We can choose those estimates that maximize the a-posteriori probability (MAP), by maximizing the posterior density with respect to the matched features and the pose. But to find such estimates, we need efficient methods for searching the objective function.

To handle this search process, Wells has considered several approaches including beam search through a tree of interpretations [51], and posterior marginal pose estimation [52]. The latter is motivated by the observation that in tree searches of the objective function of MAP model matching, hypotheses having "or" matches scored poorly in the objective function. The implication was that summing posterior probability over all matches (at a specific pose) might provide a good pose evaluator. This has proven to be the case in the experiment described in [51].

The essence of posterior marginal pose estimation is to choose the pose that maximizes the posterior probability density of the pose, given an image:

$$\hat{\beta} = \arg \max_{\beta} p(\beta | \Theta)$$

The posterior probability density of the pose is computed from the joint posterior probability on pose and match, by taking the marginal over the possible matches:

$$p(\beta | \Theta) = \sum_{\Gamma} p(\Gamma, \beta | \Theta)$$

Using Bayes' rule, the posterior marginal may be isolated as a function of the priors described above, and this leads to a convenient objective function for optimization. One can utilize the EM algorithm to provide an efficient method for optimizing this objective function, thereby leading to solutions to the pose problem. A more complete description is given the paper by Wells in these proceedings.

Wells has applied the method to several cases, include 2D point features, 2D oriented range features and linear 3d projection models. An example of recognition from visible image features is shown in Figure 5.

A second example shows the method applied to a very different type of imagery. This work was done in conjunction with Group 53 of Lincoln Labs, directed by A. Gischwendtner. In this example, the features were oriented-range features, consisting of fragments of image edge contours, augmented with a vector pointing in the direction normal to a range discontinuity, with length reflecting the inverse range at the discontinuity. Two sets of features were prepared, the "model features", and the "image features".

The object model features were derived from a synthetic range image of an M35 truck that was created using the ray tracing program associated with the BRL CAD Package [16]. The ray tracer was modified to pro-

duce range images instead of shaded images. The synthetic range image appears in the upper left of Figure 6.

In order to simulate a laser radar, the synthetic range image described above was corrupted with simulated laser radar sensor noise, using a sensor noise model that is described by Shapiro, Reinhold, and Park [40]. In this noise model, measured ranges are either valid or anomalous. Valid measurements are normally distributed, and anomalous measurements are uniformly distributed. The corrupted range image appears in Figure 6 on the right. To simulate post sensor processing, the corrupted image was "restored" via a statistical restoration method of Menon and Wells [31]. The restored range image appears in the lower position of Figure 6.

Oriented range features were extracted from the synthetic range image, for use as model features - and from the restored range image, these are called the noisy features. The features were extracted from the range images in the following manner. Range discontinuities were located by thresholding neighboring pixels, yielding range discontinuity curves. These curves were then segmented into approximately 20-pixel-long segments via a process of line segment approximation. The line segments (each representing a fragment of a range discontinuity curve) were then converted into oriented range features in the following manner. The X and Y coordinates of the feature were obtained from the mean of the pixel coordinates. The normal vector to the pixels was gotten via least-squares line fitting. The range to the feature was estimated by taking the mean of the pixel ranges on the near side of the discontinuity. This information was packaged into an oriented-range feature. The model features are shown in the fourth image of Figure 6. Each line segment represents one oriented-range feature, the ticks on the segments indicate the near side of the range discontinuity. There are 113 such object features.

The noisy features, derived from the restored range image, appear in the fifth image of Figure 6. There are 62 noisy features. Some features have been lost due to the corruption and restoration of the range image. The set of image features was prepared from the noisy features by randomly deleting half of the features, transforming the survivors according to a test pose, and adding sufficient randomly generated features so that $\frac{1}{8}$ of the features are due to the object. The 248 image features appear in the sixth image of Figure 6.

Using this data, the EM algorithm was run in a multi-resolution manner, and Figure 7 shows the convergence of the algorithm to the correct pose.

2.4 Projective Structure and Recognition

In classic projective geometry of 3D space, projective structure is typically defined by three cross-ratios using five reference points (tetrahedron of reference and a unit point) [39; 32] or, equivalently, by a tetrad of homogeneous coordinates. With such projective structure one can reconstruct the scene up to an unknown projective transformation in 3D projective space, or equivalently the camera coordinate frame may undergo an affine trans-

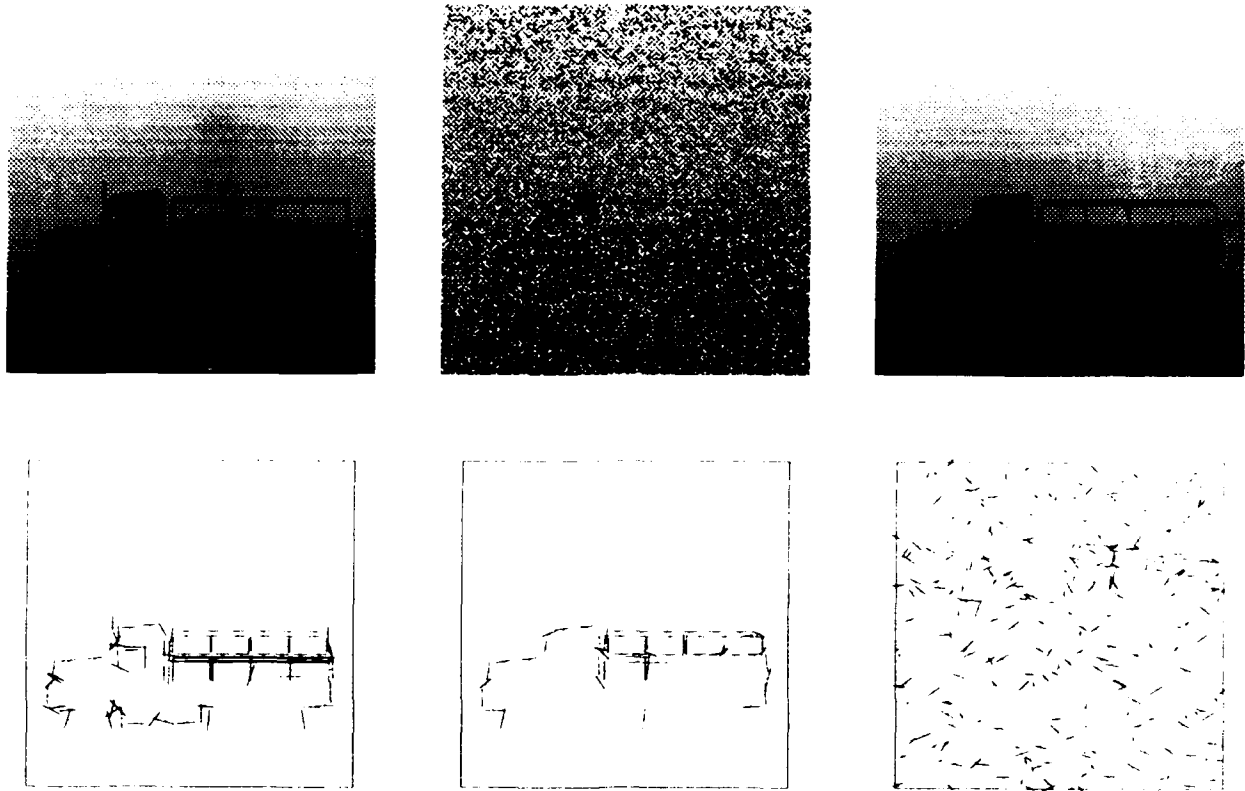


Figure 6: Top row: Synthetic range image, noisy range image, and restored range image. Bottom row: Model features, noisy features, and image features.

formation in space followed by an arbitrary projective transformation of the image plane (taking a “picture” of the image). A projective framework does not make a distinction between orthographic and perspective views and does not require camera calibration (internal camera parameters are folded into the affine transformation of the camera coordinate frame).

Our approach has several characteristics: first, our definition of projective structure relies on four scene reference points and the camera’s center (instead of five scene reference points), and is defined using a single cross-ratio (rather than three) — which means that it is obtained by some invariant function of projective coordinates. Second, the proposed projective invariant allows us to reconstruct the scene up to an unknown 3D projective transformation and, in addition, is particularly useful for recognition, that is, reconstruction of any third view becomes very simple in this framework. Thirdly, we make use of the epipoles to compute the projective invariant using only linear image-based computations ([18; 32] propose other similar techniques for using the epipoles in a linear reconstruction of homogeneous or non-homogeneous coordinates). Finally, we do not recover the camera transformation in the course of reconstruction, but instead recover the projective transformations of two faces on the tetrahedron of reference.

Taken together, projective structure can be computed from two images, perspective or orthographic, using an uncalibrated camera. The computation requires the cor-

respondences arising from four reference points and the epipoles — the latter can be recovered by a number of methods using six to eight corresponding points [29; 19; 18].

Key results here are that the structure invariant is useful for 3D reconstruction up to an unknown 3D projective transformation of the object, and for purposes of recognition via alignment by creating an equivalence class for different views of the same object. In other words, we can “re-project” an object onto any novel view given two model views in full correspondence and a small number of corresponding points between the novel view and the model views.

3 Algebraic Functions for 3D Visual Recognition

The geometric relation between different views of a 3D object can be used to recover the change in viewing geometry across views, and to recover the object’s 3D structure. The work on projective structure demonstrates that in a projective framework one can define and recover a structure invariant that is sufficient for uniquely reconstructing the object with respect to a frame of reference consisting of four scene points and the camera’s center. The location of the reference frame in space is generally unknown, and therefore the object’s structure can be reconstructed up to an unknown 3D projective transformation in space. This allows us to

work in a framework that does not make a distinction between orthographic and perspective views and does not require internal camera calibration.

The geometric relation between objects and their views can also be used for purposes of recognition. In this case one is generally not interested in recovering object structure from multiple views but instead in being able to predict the appearance of a novel view from a small number of example views ("model" views) given a small number of corresponding points between the novel view and the model views. The projective structure invariant can also be used for this purpose (see Figure ??) but it is more desirable to achieve the same result directly without going through the computation of structure (metric or non-metric) and without the reconstruction of camera geometry (transformation parameters or epipolar geometry).

In what is still an ongoing research we derive algebraic relations between image coordinates across three views (two model views and a novel view). We show here three results: first, the general result is that image coordinates across three views (perspective or orthographic) are related by a small number of third-order algebraic functions each having 11 parameters that can be recovered by linear methods. Second, if the two model views are known to be orthographic, then the algebraic functions reduce to second-order ones with only 7 parameters. Thirdly, if all three views are known to be orthographic, then the functions reduce further to first-order ones with only 4 parameters. The latter is identical to the result derived by [48; 33] known as "the linear combination of views", and thus the first two results can be viewed as an extension of the linear combination of views to perspective.

In a projective framework, five reference points (a tetrahedron and a unit point) are used for constructing a coordinate system of 3D space ([49], for example). A projection of a point P onto a plane with respect to an arbitrarily positioned center of projection (COP) and arbitrarily positioned image plane can be achieved by first mapping the reference frame such that one of the tetrahedron's vertices is aligned with the desired location of the COP, and three other vertices are coplanar with the desired image plane (in projective space five points in general position can be uniquely mapped onto any other configuration of five points in general position). The point P is then projected onto the face of the tetrahedron opposite to the COP (in homogeneous coordinate representation of space, this is achieved by an orthographic projection, see Figure 9). If we assume that there exists at least one configuration of the reference frame in which three of the reference points do not intersect any of the scene points, then it is not difficult to show (but not shown here) that the space of images we can get out of this framework are no more than perspective and orthographic images of the scene, and images of images of the scene, produced by a pin-hole camera in which the camera's coordinate frame is allowed to undergo arbitrary affine transformations in space (rather than similarity transforms used in metric structure-from-motion models).

Let x_b, y_b, z_b be the affine coordinates of a point P with respect to four of the reference points. If the fifth reference point (taken to be the COP) is not at infinity, then the observed image coordinates (x, y) can be described by an affine change of coordinates followed by a 2D projective transformation:

$$z \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = A_1 \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} + \tau,$$

for some fixed matrix A_1 and vector τ and some scale factor z (in a metric framework z would correspond to "depth"). In the case of an orthographic projection (COP at infinity and only 2D affine transformations of the image are allowed), we have:

$$\begin{pmatrix} x \\ y \\ z_b \end{pmatrix} = B_1 \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} + s,$$

for some 2D affine transformation B_1 (third row is $(0, 0, 1)$) and an ideal vector s (third coordinate of s is zero) [28; 41]. We can use these two equations to describe the transformation between image coordinates in two views across four cases: two perspective views, two orthographic views, a perspective to orthographic case, and an orthographic to perspective case. This is described below:

$$\rho' \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \rho A \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} + v. \quad (1)$$

where $\rho = z, \rho' = z'$, and A, v are general for the perspective-to-perspective case; $\rho = \rho' = \frac{1}{z_b}$, A is a 2D affine transformation and $v_3 = 0$ for the orthographic-to-orthographic case; $\rho = z, \rho' = 1$, third row of A is $(0, 0, 0)$ and $v_3 = 1$ for the perspective-to-orthographic case; $\rho = \frac{1}{z_b}, \rho' = \frac{1}{z'_b}$, and A, v are general for the orthographic-to-perspective case. Similarly, the image coordinates (x'', y'') of a third view satisfy the following relation to the first view:

$$\rho'' \begin{pmatrix} x'' \\ y'' \\ 1 \end{pmatrix} = \rho B \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} + u. \quad (2)$$

Note that ρ remains fixed regardless whether the third view is perspective or orthographic. The algebraic functions of image coordinates across three views can be derived by first eliminating ρ', ρ'' and then isolating ρ :

$$\rho = \frac{v_1 - x'v_3}{x'a_3 \cdot p - a_1 \cdot p} = \frac{v_2 - y'v_3}{y'a_3 \cdot p - a_2 \cdot p} = \frac{y'v_1 - x'v_2}{x'a_2 \cdot p - y'a_1 \cdot p},$$

where a_1, a_2, a_3 are the row vectors of A , and $p = (x, y, 1)$, $p' = (x', y', 1)$. Similarly, from equation 2 we obtain

$$\rho = \frac{u_1 - x''u_3}{x''b_3 \cdot p - b_1 \cdot p} = \frac{u_2 - y''u_3}{y''b_3 \cdot p - b_2 \cdot p} = \frac{y''u_1 - x''u_2}{x''b_2 \cdot p - y''b_1 \cdot p},$$

where b_1, b_2, b_3 are the row vectors of B , and $p'' = (x'', y'', 1)$. These two equations lead to nine algebraic functions of image coordinates across three views.

For example, the first two terms lead to the function $F(x'', x', x, y) = 0$ of the form:

$$x''(v_1 b_3 \cdot p - u_3 a_1 \cdot p) + x'x'(u_3 a_3 \cdot p - v_3 b_3 \cdot p) + x'(v_3 b_1 \cdot p - u_1 a_3 \cdot p) + (u_1 a_1 \cdot p - v_1 b_1 \cdot p) = 0. (3)$$

Note that the bracketed terms are first-order polynomials of x and y with fixed coefficients (depending only on parameters of camera transformations). In other words, equation 3 is a third-order algebraic function of the form:

$$x''(\alpha_1 x + \alpha_2 y + \alpha_3) + x'x'(\alpha_4 x + \alpha_5 y + \alpha_6) + x'(\alpha_7 x + \alpha_8 y + \alpha_9) + \alpha_{10} x + \alpha_{11} y + \alpha_{12} = 0, (4)$$

where the coefficients α_j , $j = 1, \dots, 12$, are fixed constants. Note that the constants can be recovered by linear methods by observing 11 corresponding points across the three views (more than 11 points can be used for a least-squares solution). Then, for any additional point (x, y) whose correspondence in the second image is known (x', y') , we can recover the corresponding x -component x'' in the third view by substitution in equation 4. In a similar fashion we can recover the y -component y'' by using one of the other functions, for example:

$$y''(\beta_1 x + \beta_2 y + \beta_3) + y'y'(\beta_4 x + \beta_5 y + \beta_6) + y'(\beta_7 x + \beta_8 y + \beta_9) + \beta_{10} x + \beta_{11} y + \beta_{12} = 0. (5)$$

The solution for x'' , y'' is unique provided that v_1, v_3 do not vanish simultaneously, or u_1, u_3 do not vanish simultaneously. These singular cases apply only to the two functions above, and one can show that from the nine functions we can always find two that are not singular under any viewing transformations that takes place between the three views. The process of generating a novel view can be easily accomplished without the need to explicitly recover structure, camera transformation or epipolar geometry — with the price of using more than the minimal eight points that are required by less direct methods.

The algebraic functions derived so far are general in the sense that the scene is allowed to undergo general 3D projective transformation in space. Reduced lower order functions can be derived under more restricted situations. For example, the third order component of these functions vanishes when $v_3 = u_3 = 0$ (see Equation 3). This situation arises, for example, when the views are taken by a camera moving along a base line perpendicular to the optical axis. One observes, as a result, that this situation is intrinsically more stable (errors in correspondence multiply to a second-order rather than to a third-order) than the general case — an observation experimentally made by [7; 5].

Other results can be obtained by assuming that some of the views are orthographic. This is especially important in the context of achieving recognition via alignment: since the two model views are taken only once (and offline), we may as well use a tele-lense for producing orthographic views. In this case we substitute $v_3 = 0$ and $u_3 \cdot p = 1$ in Equation 3 and obtain a second-order function with only 7 free parameters which has the form:

$$x''(\alpha_1 x + \alpha_2 y + \alpha_3) + \alpha_4 x'x' + \alpha_5 x' + \alpha_6 x + \alpha_7 y + \alpha_8 = 0. (6)$$

for some fixed constants α_j , $j = 1, \dots, 8$. As a result, we can generate novel views (perspective and orthographic) by observing only 7 corresponding points across the three views. This result is both direct (avoiding structure and motion) and requires less than eight points (the minimal under general conditions using linear methods). For instance, with other tools we do not have an easy way for making use of the fact that the two model views are orthographic — because the determination of the epipoles and epipolar geometry between a perspective and an orthographic view still requires eight points in general.

Finally, it is easy to see what happens when all three views are orthographic. In this case we have also $u_3 = 0$ and $b_3 \cdot p = 1$, and thus Equation 3 reduces to a first-order function, with only 4 free parameters, of the form:

$$\alpha_1 x'' + \alpha_2 x' + \alpha_3 x + \alpha_4 y + \alpha_5 = 0, (7)$$

for some fixed constants α_j , $j = 1, \dots, 5$. This is identical to the "linear combination of views" result [48; 33], stating that under the orthographic assumption an arbitrary view can be generated by applying certain linear combinations to the image coordinates of two model views.

To summarize, we have shown that it is possible to represent views as a function of image coordinates of two other views. In the general projective case, the image coordinates of three views are connected via third-order algebraic functions with 11 free parameters. More restrictive cases (but applicable in the context of visual recognition) reduce these functions to second-order with 7 free parameters and to first-order with 4 free-parameters depending on whether two or all the views are assumed to be orthographic. The immediate application of these results are in the context of visual recognition via alignment (especially the 7-point result), but other applications may also be possible. For example, the general result (Equation 4) may be useful in the context of model-based image compression. In this case the number of corresponding points required for reconstructing novel views is not of critical importance whereas robustness and simplicity are more of a concern. The 22 parameters required for reconstructing a novel view can be recovered by many points in a least-squares fashion, but the receiver eventually requires only the parameters and not the corresponding points.

3.1 Recognition and Structure from one 2D Model View

According to the 1.5 views theorem [33; 6] recognition of a specific 3D object (defined in terms of pointwise features) from a novel 2D view can be achieved from at least two 2D model views (in the data basis, for each object, for orthographic projection). Poggio and Vetter studied how recognition can be achieved from a single 2D model view. The basic idea is to exploit transformations that are specific for the object class corresponding to the object — and that may be known a priori or may be learned from views of other "prototypical" objects of the same class — to generate new model views from the only one available. Their work divides in two distinct parts. In the first part, they discuss how to exploit prior

knowledge of an object's symmetry. They prove that for any bilaterally symmetric 3D object one non-accidental 2D model view is sufficient for recognition. They also prove that for bilaterally symmetric objects the correspondence of four points between two views determines the correspondence of **all** other points. Symmetries of higher order allow the recovery of structure from one 2D view. In the second part of their work, Poggio and Vetter study a very simple type of object classes called **linear object classes**. Linear transformations can be learned exactly from a small set of examples in the case of linear object classes and used to produce new views of an object from a single view. More recently Vetter, Poggio and Buelthoff have provided psychophysical support for the hypothesis that the human visual system exploits symmetry of 3D objects for better generalization from a few views.

3.2 Face Recognition: Features versus Templates

Over the last twenty years several different techniques have been proposed for computer recognition of human faces. Poggio in collaboration with R. Brunelli at IRST compared two simple but general strategies on a common database (frontal images of faces of 47 people, 26 males and 21 females, four images per person). They have developed and implemented two new algorithms, the first one based on the computation of a set of geometrical features, such as nose width and length, mouth position and chin shape, and the second one based on almost-grey-level template matching. The results obtained on the testing sets, about 90% correct recognition using geometrical features and perfect recognition using template matching, favour their implementation of the template matching approach. Present work aims to extend the system to deal with arbitrary poses and expressions of the face.

4 Navigation

Complementary to our work on object recognition, we have also investigated issues and methods in navigation. One such method has built directly on our earlier work in recognition by Linear Combinations, and is reported in a separate article by Basri in these proceedings.

A second approach to navigation has been part of a broader research project, executed by Ian Horswill. The problem under consideration is the development of simple real-time vision algorithms suitable for low-cost computer systems such as personal computers. The specific goals are to develop (a) simple vision algorithms useful for problems such as robot navigation and interacting with people, (b) a theoretical framework for analyzing such systems, and (c) a concrete implementation demonstrating the algorithms in a robot which gives primitive "tours" of the seventh floor of the laboratory.

This follows from the motivation of making vision cheap as a necessary part of making it useful. For vision technology to be used routinely in construction and manufacturing equipment, consumer electronics products, automobiles, etc., both design costs and unit costs must be brought down to levels comparable with the

product into which they are to be incorporated. Million dollar supercomputers, or even twenty thousand dollar workstations are simply inappropriate for mass-produced products intended to cost less than the computer.

Horswill's approach is one of situated agents, whereby vision systems can be made much simpler and cheaper by specializing them to a specific task and environment. A task-specific system need only extract the specific information needed to perform the task. As well, a task provides performance constraints that can simplify the design process by allowing the use of approximate solutions which might not be appropriate for all tasks. A system specialized to its environment can take advantage of domain knowledge which can simplify computational problems. For example, a complete stereo system can sometimes be replaced by a system which uses height in the image plane to determine rough distance from the agent.

A critical problem in developing these systems is the reusability of components. It is important that we be able to apply experience gained in designing one system to the design of other systems. For this reason, tasks and environments must be analyzed at a theoretical level so as to make explicit the ways in which they simplify computational problems.

Low cost task-oriented vision systems could significantly improve the performance and flexibility of autonomous systems and reduce their cost. Such systems have applications in transportation, surveillance, construction, manufacturing, space applications, and hazardous operations. Low cost vision technology could also be extremely valuable in consumer electronics. Tracking systems could be incorporated into camcorders or surveillance systems. Face recognition, person detection, stereo and motion algorithms could be incorporated into intelligent nightscopes and binoculars. Low unit costs would be particularly important in this area.

To date, we have developed systems for tracking and following moving objects, detecting obstacles, proximity detection using stereo (see Figures 10 and 11), following corridors, and recognizing nods and shakes of the head. All systems run in real time on inexpensive computers such as Macintoshes. All systems use very low resolution processing (64×48 or less). A number of optimizations are shared between two or more systems, suggesting that some amount of recycling of design time is possible.

The particular prototype system is an indoor navigation system for a mobile robot that runs at 15 frames per second on stock hardware. A computer equivalent to the one on board the robot can be purchased for \$3-4K. At present, the robot is capable of following corridors, avoiding obstacles, recognizing corridor junctions, navigating from point to point, and detecting the presence of people. The corridor follower is extremely well tested, having seen hundreds of hours of service. The other capabilities are newer and have not yet been fully evaluated. An article by Horswill in these proceedings further describes the approach.

5 Early Vision Modules

Although a primary focus has been on recognition and navigation, we continue to develop methods for early processing of visual information.

5.1 Optical Flow from 1D Correlation: Application to a simple Time-To-Crash Detector

Ancona (C/SATA, Italy) and Poggio have shown that a new technique exploiting 1D correlation of 2D or even 1D patches between successive frames may be sufficient to compute a satisfactory estimation of the optical flow field. The algorithm is well-suited to VLSI implementations and a patent application is being filed by MIT and C/SATA. The sparse measurements provided by the technique can be used to compute qualitative properties of the flow for a number of different visual tasks. In particular, they also showed how to combine the 1D correlation technique with a scheme for detecting expansion or rotation [37] in a simple algorithm which also suggests interesting biological implications. The algorithm provides a rough estimate of time-to-crash. It was tested with good results on real image sequences.

5.2 Sensor Integration

Clay Thompson has recently completed a Ph.D. thesis that considers the problem of fusing two computer vision methods, using variational methods. The example algorithms solve the *photo-topography problem*; that is, the algorithms seek to determine planet topography given two images taken from two different locations with two different lighting conditions. The algorithms each employ a single cost function that combines the computer vision methods of shape-from-shading and stereo in different ways. The algorithms are closely coupled and take into account all the constraints of the photo-topography problem. One such algorithm, the *z-only* algorithm, can accurately and robustly estimate the height of a surface from two given images.

5.3 Simple Region Features

Feature-based methods for recovering the motion of a camera from a sequence of images have suffered from the inability of the early vision processes to provide dense, robust features. Typically, the features, such as Canny edges, are very sparse in each image. Furthermore, most features are unreliable in the sense that they often disappear from one frame to the next. Similarly, sporadic features often appear that are not associated with any object in the scene. To address these problems, Ron Chaney has developed a framework for early vision processing that leads to a dense, robust set of features. The early vision framework is based on the interpretation of the Laplacian of Gaussian (LoG) filter as a matched filter for features of a particular size as well as an edge locator. An object in the image that has roughly the optimum width associated with a particular LoG filter will typically be nearly surrounded by the zero-crossings of the LoG filter. Hence, a naive approach would be to take the regions bounded by the zero-crossings of the LoG filter as the set of features. Of course, the regions associated with nearby objects in the image tend to merge

or blend together. To alleviate this problem, we introduce a stable, robust decomposition of such regions into their salient subparts. These subregions, called *simple region features*, serve as the feature set for higher level processing. The decomposition is based on the medial axis skeleton of the region. Each subregion corresponds to a portion of a branch of the skeleton; each branch is divided at positions where the distance from the skeleton to the bounding contour is minimized. To facilitate the computation of the decomposition, a novel scale-space is introduced for contours and the medial axis skeleton. The scale-space is parametric with the complexity of the contour or the skeleton. The complexity measure of the skeleton is the number of branches. A related complexity measure of a contour is the number of extrema of curvature of the contour. This leads to a complexity scale-space for the region decomposition. The result of the early vision process is the set of simple region features for each frame. These features are dense, stable, and robust. To demonstrate the utility of the early vision process, we present a relatively simple motion and structure-from-motion algorithm based on tracking simple region features at multiple resolutions of the LoG filter.

5.4 Calibration

Computing relative orientation is an important problem for calculating depth from binocular stereo and for determining general camera motion. Lisa Dron has been exploring methods for establishing the complete design of a small, autonomous system with specialized VLSI hardware for computing relative orientation in real-time [14]. Such a system would be suitable for mounting on mobile or remote platforms that cannot be tethered to a computer and for which the size, weight and power consumption of the components are critical factors.

There are two parts to this work. The first is theoretical and involves developing and adapting algorithms for finding point correspondences and solving the motion equations which are robust as well as simple enough to be easily implemented in hardware. The second part is engineering and involves the design, fabrication and test of prototype chips for the specialized processors which will be used to find the point correspondences. Two separate processors are needed: one which computes a binary edge map from the input image data, and the other which determines translational offsets between patches of the edge maps from two different images. Fabrication of these circuits is done through MOSIS.

In support of such work, Dron has already developed the edge detection algorithm known as the multi-scale veto (MSV) method [15]. During the past year, she has completed the design of a two-dimensional processor which combines CCD and CMOS technology to implement the MSV algorithm. A test chip containing a 4x4 two-dimensional array has been fabricated and is currently being tested. Algorithms have been developed both to perform matching with the binary edge signals produced by the MSV chip, and to solve the motion equations with a least-squares method suitable for implementation on a programmable digital microprocessor. In

addition, Dron has developed a least-squares algorithm to determine the internal camera calibration parameters, which are required in order to compute motion from a set of point matches, using a sequence of images for which the translational motion is known. A preliminary design, comprising both analog and digital components, has been completed for the second processor which will compute point correspondences from the edge maps, and have sent out for fabrication a set of test structures which will form the basis of the matching circuit.

5.5 Other Calibration Methods

In related work, Gideon Stein has developed a simple method for internal camera calibration for computer vision systems. It is intended for use with medium to wide angle camera lenses. With modification it can be used for longer focal lengths. This method is based on tracking image features through a sequence of images while the camera undergoes pure rotation. This method does not require a special calibration object. The location of the features relative to the camera or to each other need not be known. It is only required that the features can be located accurately in the image. This method can therefore be used both for laboratory calibration and for self calibration in autonomous robots working in unstructured environments. The method works when features can be located to single pixel accuracy but subpixel accuracy should be used if available.

In the basic method the camera is mounted on a rotary stage so that the angle of rotation can be measured accurately and the axis of rotation is constant. A set of image pairs is used with various angular displacements. If the internal camera parameters and axis of rotation were known one could predict where the feature points from one image will appear in the second image of the pair. If there is an error in the internal camera parameters the features in the second image will not coincide with the feature locations computed using the first image. One can then perform a nonlinear search for camera parameters that minimize the sum of distances between the feature points in second image in each pair and those computed from the first image in each pair, summed over all the pairs.

The need to accurately measure the angular displacements can be eliminated by rotating the camera through a complete circle while taking an overlapping sequence of images and using the constraint that the sum of the angles must equal 360 degrees.

The closer the feature objects are located to the camera the more important it is that the camera does not undergo any translation during the rotation. A method is described which enables one to ensure that the axis of rotation passes sufficiently close to the center of projection (or front nodal point in a thick lens) to obtain accurate results.

Stein shows that by constraining the possible motions of the camera in a simple manner it is possible to devise a robust calibration technique that works in practice with real images. Experimental results show that focal length, aspect ratio and lens distortion parameters can be found to within a fraction of a percent. The location of the

principal point and the location of the center of radial distortion can each be found to within a few pixels.

In addition to the first method a second method of calibration is presented. This method uses simple geometric objects such as spheres and straight lines to find, first the aspect ratio, then the lens distortion parameters and finally the principal point and focal length. Calibration is performed using both methods and the results compared.

6 Other Topics

Two other recently complete theses, reported in detail in earlier reports are Steve White's work in highly accurate representations for early vision, especially edges and stereo disparities [54], and Subirana's work on recognition and representation of flexible objects [45].

6.1 Median Window Filtering for Multi-dimensional Image Attributes

Median window filtering is a simple non-linear technique for reducing image noise while preserving sharp discontinuities. It works by replacing the current value of each image pixel with the median value of the pixel's local neighborhood. Although the technique has been extensively used for smoothing scalar image data like grey-level intensities, little work is known about median filtering in multi-dimensional data domains like image color, image texture and motion fields. Perhaps this is because the sample median is an ill-defined concept for multi-dimensional quantities. Recently, Sung has proposed a novel interpretation of the median concept for multi-dimensional metric spaces. The interpretation follows from a mathematical property of the scalar median, and the basic idea is to similarly define the multi-dimensional median as the sample member that minimizes a mean absolute error term. Sung implemented a multi-dimensional median filtering algorithm for color images and showed that the operation indeed preserves edges while reducing noise. He has also mathematically derived that in the best case, the smoothing performance of multi-dimensional median filtering is comparable to that of local averaging. More recently, he has also developed algorithms for approximating multi-dimensional medians that run in linear time with respect to data dimension and sample size.

6.2 Statistical Uniformity Based Region Finding

Over the past twenty to thirty years, a number of different techniques have been proposed for segmenting images into piecewise uniform regions. Like many other computer vision tasks, most of these techniques contain at least a few thresholds and operating parameters whose values are crucial for producing reasonable results. Often however, these values are determined either empirically or by guess. Kah Kay Sung has explored an alternative approach to the threshold selection problem for a simple but fairly general class of uniformity based region finding paradigms. He proposed a statistical formulation for uniformity based region finding as a series of "confidence" and "significance" tests, where each test roughly

corresponds to a decision procedure in the original region finding paradigm. The main advantage of his approach is that it replaces typical region finding thresholds and parameters with a new set of confidence and significance thresholds and parameters that gives greater insight to the system's pertinent characteristics. A color region algorithm, based on his formulation, was implemented on the parallel Connection Machine.

7 Learning

Under separate contract, Tomaso Poggio and colleagues have been developing techniques for the application of learning methods to vision problems. In particular, building on extensive earlier work by Poggio and collaborators on the use of Generalized Radial Basis Functions, they have been developing learning methods for use in object recognition and computer graphics.

8 VLSI

Under separate contract Berthold Horn and colleagues have continued to develop VLSI implementations of low level visual algorithms.

8.1 The Focus of Expansion Chip

The problem that this chip solves is that of computing the direction towards which a camera is moving, based on the time-varying image it receives. There is no restriction on the shapes of the surfaces in the environment; only an assumption that the imaged surfaces have some texture, that is, spatial variations in reflectance. It is also assumed that the camera is stabilized so that there is no rotational motion.

Once the translational motion has been determined, it is possible to estimate distances to points in the scene being imaged. While there is an ambiguity in scale, since multiplying both distances and speed by some constant factor does not change the time-varying image, it is possible to estimate the ratio of distance to speed. This allows one to estimate the time-to-collision between the camera and objects in the scene.

Applications for such a device include systems warning of imminent collision, obstacle avoidance in mobile robotics, and aids for the blind.

The projection of the translational motion vector into the image is called the focus of expansion (FOE). It is the image of the point towards which the camera is moving, and the point from which other image points appear to be receding.

The method used is based on least squares analysis - that is, find the point in the image that best fits the observed time variations in brightness. The quantity minimized is the sum of squares of the differences at every picture cell between the observed time variation of brightness and that predicted, given the assumed position of the FOE and the observed spatial variations of brightness.

The minimization is not straightforward, because the relationship between the brightness derivatives depends on distance to the surface being imaged and that distance is not only unknown, but varies from picture cell

to picture cell. It turns out that so-called stationary points, where brightness is constant (instantaneously), play a critical role. If there were no measurement errors, quantization effects or noise, then the FOE would be at the intersection of the tangents to the iso-brightness contours at these stationary points.

In practice, image brightness derivatives are hard to estimate accurately given that the image itself is quite noisy. Hence the intersections of tangents from different stationary points may be quite scattered. Reliable results can nevertheless be obtained if the image contains many stationary points and the point is found that has the least weighted sum of squares of perpendicular distances from the tangents at the stationary points.

This method was chosen from amongst a group of competing approaches by considering both simulation results of these methods and constraints of what can reasonably be built in analog VLSI.

The amount of computation for every picture cell (including a number of multiplications) is such that it is not feasible today to perform the task in a totally unclocked manner with the processing done at each picture cell. Instead a row parallel scheme has been decided upon where each row of the image has a single processor.

The first chip has been made by MOSIS and tested. Minor revisions are being made.

8.2 System for recovering motion with respect to a planar surface

A problem in motion vision that is somewhat more difficult than that of recovering the focus of expansion is that of recovering both translational and rotational components of motion of a camera from the time-varying image. Presently there is no simple robust method for solving this problem in general, but methods are known in the special case that the surface being viewed is planar.

Applications for such a system include landing a vehicle on a planar surface and station keeping of a submersible vehicle above the ocean floor. Also, such a system could be used to recover the motion of a person by aiming a camera at the flat ground in front of the person. The motion estimates obtained from such a downward looking camera could then be used to interpret the time-varying image from a second camera aimed directly forward. The resulting system could be an aid for the blind that warns them of obstacles - even those that do not have a support directly below - such as signs hanging from beams supported off on the side.

Here also the proposed method involves a least squares approach, although it is now considerably more complex than in the case of simple translational motion. It is known, for example, that there is an ambiguity in that two quite different motions, and corresponding different surface orientations, can yield the same time-varying image.

Detailed design will have to await the results of extensive simulations.

8.3 Analog Circuits

John Harris and Prof. Poggio are studying analog implementations of vision and learning algorithms. They are interested in analog models because these models provide a novel mechanism for understanding and developing algorithms. Experimentation with these continuous-time nonlinear circuits facilitates algorithm intuition and leads to fundamental insights. Powerful analog algorithms thus developed will prove useful even if a researcher is limited to simulating the analog hardware on a digital computer. In addition, biology has motivated some of the circuits and, conversely, some of the VLSI modules may help develop a better intuition for solutions that biology has found for the same class of early vision problems.

A real-world vision system must be adaptive in order to operate in a unconstrained environment. The system must be smart enough to deal with such nonidealities as changing light conditions or slight variations between components. For example, the thresholds for detection of edges in edge detectors should dynamically change with the brightness of objects in the scene. Or the appropriate space constant of resistive network could be dynamically determined by an estimate of the noise in the input signal. Analog hardware allows for adaptation in many instances by relying on basic physics to perform the necessary computations. One specific project under implementation is the time-to-contact motion sensor proposed by Poggio (1991) and Poggio & Ancona (1992). This sensor combines the outputs of 1D motion correlation sensors to produce an estimate of the amount of time until crash (assuming constant velocity). The small, low-power implementation will be useful as a crash-warning sensor for robots or automobiles.

References

- [1] T. Alter. Fast and robust 3d recognition by alignment. Master's thesis, MIT, 1992.
- [2] T. D. Alter. 3d pose from 3 corresponding points under weak-perspective projection. Memo 1378, MIT AI Lab, 1992.
- [3] T. D. Alter and W. E. L. Grimson. Fast and robust 3d recognition by alignment. In *Proc. Fourth Inter. Conf. Computer Vision*, 1993.
- [4] H.S. Baird. *Model-Based Image Matching Using Location*. MIT Press, Cambridge, 1985.
- [5] H.H. Baker and R.C. Bolles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. *Int. J. Comp. Vision*, 3, 1989.
- [6] R. Basri and S. Ullman. The alignment of objects with smooth surfaces. In *Second Int. Conf. Comp. Vision*, pages 482-488, 1988.
- [7] R.C. Bolles and H.H. Baker. Epipolar-plane image analysis: A technique for analyzing motion sequences. In *IEEE Proc. of the 3rd Workshop on Computer Vision: Representation and Control*, Bellaire, MI, October 1985.
- [8] T.M. Breuel. Fast recognition using adaptive subdivisions of transformation space. In *IEEE Conf. Comp. Vis. Patt. Recog.*, 1992a.
- [9] T.M. Breuel. View-based recognition. In *IAPR Workshop on Machine Vision Applications*, 1992b.
- [10] T.A. Cass. *Polynomial-Time Geometric Matching for Object Recognition*. PhD thesis, MIT, 1992.
- [11] Ronald D. Chaney. Analytical representation of contours. AI Memo 1392, MIT Artificial Intelligence Laboratory, 1992.
- [12] Ronald D. Chaney. Computation of the medial axis skeleton at multiple complexities. In *Proceedings of Intelligent Robots and Computer Vision XI: Algorithms, Techniques, and Active Vision*, Boston, MA, November 1992.
- [13] D.T. Clemens. *Region-Based Feature Interpretation for Model-Based Recognition*. PhD thesis, MIT, 1991.
- [14] L. Dron. System-level design of specialized vlsi hardware for computing relative orientation. In *Proc. IEEE Workshop on Applications of Computer Vision*, pages 128-135, 1992a.
- [15] L. Dron. The multi-scale veto model: A two-stage analog network for edge detection and image reconstruction. Memo 1320, MIT AI Lab, 1992b.
- [16] P. Dykstra and M.J. Muuss. The brl cad package: an overview. In *Proc. Fourth USENIX Computer Graphics Workshop*, pages 73-80, 1987.
- [17] S. Edelman and T. Poggio. Bringing the grandmother back into the picture: a memory-based view of object recognition. *Int. J. Pattern Recog. Artif. Intell.*, 6:37-61, 1992.
- [18] O.D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Eur. Conf. Comp. Vision*, pages 563-578, 1992.
- [19] O.D. Faugeras and S. Maybank. Motion from point matches: Multiplicity of solutions. *Int. J. Comp. Vision*, 4:225-246, 1990.
- [20] W. E. L. Grimson, D. P. Huttenlocher, and T. D. Alter. Recognizing 3d objects from 2d images: An error analysis. In *Proc. IEEE Conf. Computer Vision Pat. Rec.*, 1992.
- [21] W. E. L. Grimson, D. P. Huttenlocher, and D. W. Jacobs. A study of affine matching with bounded sensor error. In *Second European Conf. on Computer Vision*, pages 291-306, 1992.
- [22] W.E.L. Grimson. *Object Recognition by Computer: The role of geometric constraints*. MIT Press, Cambridge, 1990.
- [23] J.G. Harris. An analog network for continuous-time segmentation. *Internat. Journal of Comp. Vision*, 1993.
- [24] D.P. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proc. Int. Conf. Comp. Vision*, pages 102-111, 1987.
- [25] D.P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *Int. J. Comp. Vision*, 5(2):195-212, 1990.
- [26] D.W. Jacobs. Space efficient 3d model indexing. In *IEEE Conf. Comp. Vis. Patt. Recog.*, pages 439-444, 1992.
- [27] D.W. Jacobs. *Recognizing 3D Objects Using 2D Images*. PhD thesis, MIT, 1992a.
- [28] J.J. Koenderink and A.J. Van Doorn. Affine structure from motion. *J. Opt. Soc. Amer.*, 8(2):377-385, 1991.

- [29] C.H. Lee. Structure and motion from two perspective views via planar patch. In *Int. Conf. Comp. Vision*, pages 158–164, 1998.
- [30] S. Liu and J.G. Harris. Dynamic wires: an analog vlsi model for object processing. *Internat. Journal of Comp. Vision.*, 8:231–239, 1992.
- [31] M. Menon and W.M. Wells III. Massively parallel image restoration. In *Proc. Int. Joint Conf. Neur. Nets*, 1990.
- [32] R. Mohr, L. Quan, F. Veillon, and B. Boufama. Relative 3D reconstruction using multiple uncalibrated images. Memo RT 84-IMAG, LIFIA – IRIMAG, 1992.
- [33] T. Poggio. 3d object recognition: On a result by Basri and Ullman. Tech Report 9005-03, IRST, Povo, Italy, 1990.
- [34] T. Poggio and R. Brunelli. A novel approach to graphics. Memo 1354, MIT AI Lab, 1992.
- [35] T. Poggio, S. Edelman, and M. Fahle. Learning of visual modules from examples: a framework for understanding adaptive visual performance. *CVGIP: Image Understanding*, 56:22–30, 1992.
- [36] T. Poggio, M. Fahle, and S. Edelman. Fast perceptual learning in visual hyperacuity. *Science*, 256:1018–1021, 1992.
- [37] T. Poggio, A. Verri, and V. Torre. Greens theorems and qualitative properties of the optical flow. Memo 1289, MIT AI Lab, 1991.
- [38] T. Poggio and T. Vetter. Recognition and structure from one 2-d model view: Some observations on prototypes, object classes and symmetries. Memo 1347, MIT AI Lab, 1992.
- [39] J.G. Semple and G.T. Kneebone. *Algebraic Projective Geometry*. Oxford, Clarendon Press, 1952.
- [40] J.H. Shapiro, R.W. Reinhold, and D. Park. Performance analyses for peak-detecting laser radars. In *Proc. SPIE*, pages 38–56, 1986.
- [41] A. Shashua. Correspondence and affine shape from two orthographic views: Motion and recognition. Memo 1327, MIT AI Lab, 1991.
- [42] A. Shashua. *Geometry and Photometry in 3D visual recognition*. PhD thesis, M.I.T Artificial Intelligence Laboratory, AI-TR-1401, November 1992.
- [43] A. Shashua. Illumination and view position in 3D visual recognition. In S.J. Hanson J.E. Moody and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 404–411. San Mateo, CA: Morgan Kaufmann Publishers, 1992. Proceedings of the fourth annual conference NIPS, Dec. 1991, Denver, CO.
- [44] A. Shashua. Projective invariant from two views: Applications in reconstruction of affine or projective structure and 3d visual recognition. In *Int. Conf. Comp. Vis.*, Berlin, Germany, May 1993.
- [45] B. Subirana-Vilanova. *Visual Recognition of Non-rigid Objects and Perceptual Organization*. PhD thesis, MIT, 1993.
- [46] T. Syeda-Mahmood. *A Computational Model of Visual Attention*. PhD thesis, MIT, 1993.
- [47] T.F. Syeda-Mahmood. Data and model-driven selection using color regions. In *Eur. Conf. Comp. Vis.*, pages 321–327, 1992.
- [48] S. Ullman and R. Basri. Recognition by linear combination of models. *IEEE Trans. PAMI*, PAMI-13:992–1006, 1991. Also in M.I.T AI Memo 1052, 1989.
- [49] O. Veblen and J.W. Young. *Projective Geometry, Vol. 1*. Ginn and Company, 1910.
- [50] T. Vetter, T. Poggio, and H. Bülthoff. 3d object recognition: Symmetry and virtual views. Memo 1409, MIT AI Lab, 1992.
- [51] W.M. Wells. Map model matching. In *Proceedings IEEE Conf. Comp. Vis. Patt. Recog.*, pages 486–492, 1991.
- [52] W.M. Wells. Posterior marginal pose estimation. In *Proceedings of the Image Understanding Workshop*, 1992a.
- [53] W.M. Wells. *Statistical Object Recognition*. PhD thesis, MIT, 1992b.
- [54] S.J. White. *Displacement and Disparity Representations in Early Vision*. PhD thesis, MIT, 1992.
- [55] J.L. Wyatt, C. Keast, M. Seidel, D. Standley, B. Horn, T. Knight, C. Sodini, H.-S. Lee, and T. Poggio. Analog vlsi systems for image acquisition and fast early vision processing. *Intl. J. Computer Vision*, 8:217–230, 1992.

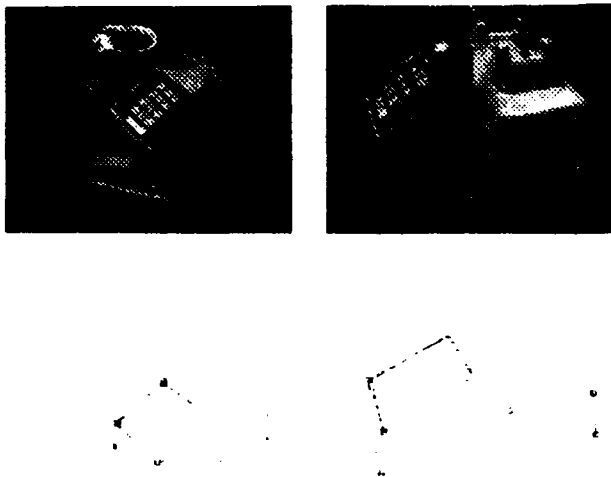


Figure 2: Examples of recognition using affine indexing. In the hypothesized solutions shown below each example, edges are dotted lines, projected line segments are lines, circles represent image corners in the match, and squares show the location of the projected model corners. Matching is based on idea that a model is stored as a line in $\alpha - \beta$ space, and an image is simply a lookup in that space, based on the affine coordinates of a set of points. Shown here are the retrieved models from such a lookup.

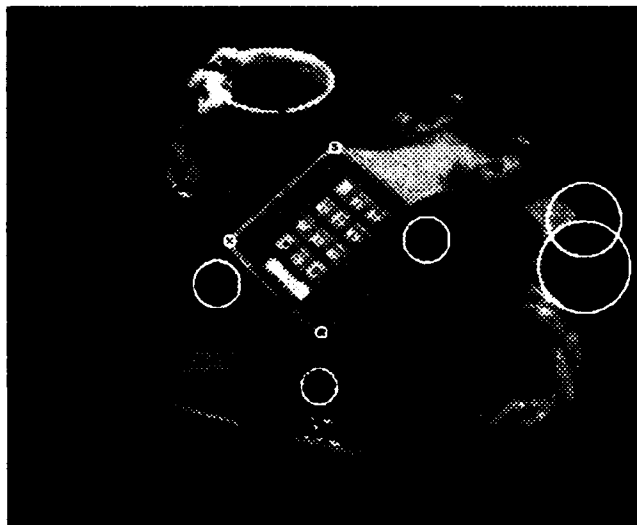


Figure 3: Example of alignment with an image. The three small circles show the uncertainty about the matched basis points. The other circles show the formal error regions in which to search for matches for other point features corresponding to the corners of the phone.

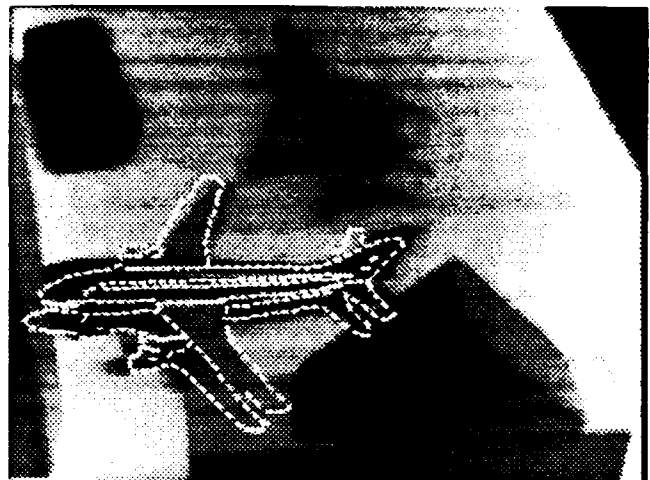


Figure 4: Examples of recognition using the RAST algorithm.



Figure 5: Examples of recognition using statistical optimization, based on matching image edge segments. Solutions are overlaid on original image.

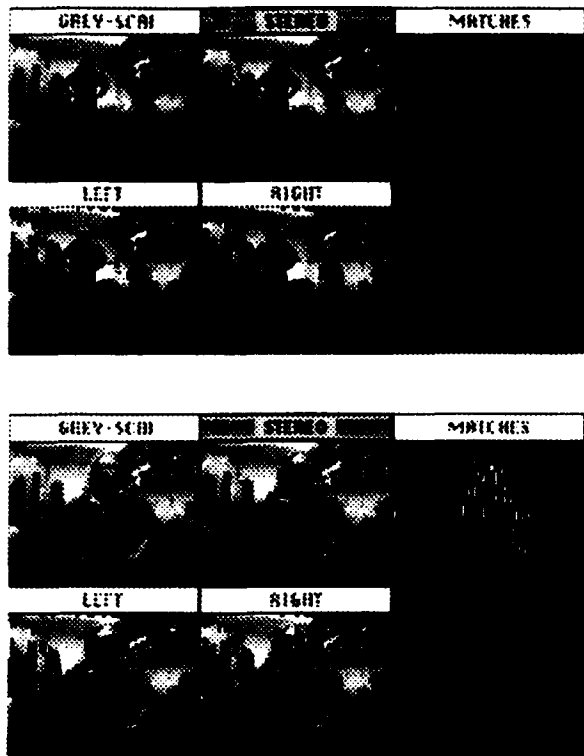


Figure 10: Stereo Proximity Detector. Performance of the system in an unmodified office environment. "Grey-scale" is the original sampled image from the left camera, "left" and "right" are the respective images overlaid with derived edges, "matches" is the set of edges matched between images at disparity 2, and "stereo" is the left grey scale image overlaid with matched edges.

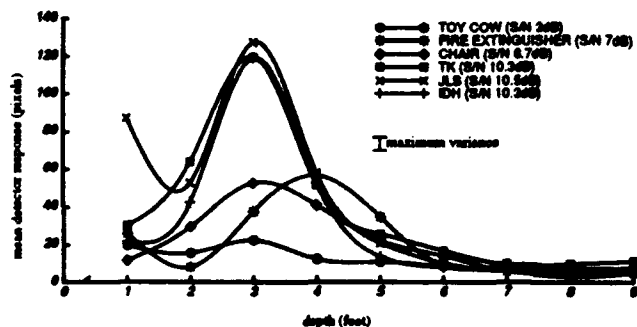


Figure 11: Depth-tuning curves and peak S/N ratios for various objects. S/N ratios for objects are measured as the ratio of peak response to that object over the noise level (12 pixels). Images are 64×48 pixels, cameras have 110 degree fields of view with a baseline of 65mm. Objects IDH, JLS, and TK are people. Each object was tested in nine positions of varying depth for 20 trials each. The largest variance was 6.9 pixels. Readings past 7 feet are entirely matching noise. The noise level was measured as the largest such noise value observed over all trials for all data points. The background was a cluttered office.

Image Understanding Research at Columbia University

Terrance E. Boult, Peter K. Allen, John R. Kender, Shree K. Nayar*

Center for Research in Intelligent Systems, Department of Computer Science,
Columbia University New York, N.Y. 10027, U.S.A.

Abstract

This is an overview of IU research in image understanding at Columbia's Center for Research in Intelligent Systems since the 1992 image understanding workshop. It reviews our work on the following topics:

1. Physics-Based Vision
 - (a) Generalization of Lambertian Model
 - (b) Polarization-Based Techniques
2. Recognition and Learning
 - (a) Reflectance-Based Recognition
 - (b) Object Recognition using BSP trees
 - (c) Shape Recovery
 - (d) Learning and Recognition from Appearance
3. Sensor and Illumination Planning
 - (a) Planning in Active Environments
 - (b) Illumination Planning
 - (c) Modeling Uncertainty
 - (d) IUE sensors
4. Qualitative Vision
 - (a) Alignment using Uncalibrated Cameras
 - (b) Topological Navigation
 - (c) Qualitative Spatial Description
5. Real-Time Vision
 - (a) Real-Time Polarization Computation
 - (b) Real-Time Image Warping
 - (c) Real-Time Tracking
 - (d) Real-Time Detail-Preserving Smoothing

As you can see our research covers the full span of computer vision, from low-level image processing to complete systems integration of vision and robotics. Here we present only appetizers; for the full course the hungry reader should consult the research papers referenced in this overview.

*This work supported in part by DARPA contract DACA-76-92-C-007. Numerous other agencies and companies have also supported parts of this research.

1 Physics-Based Vision

The topic of physics-based vision is enjoying a resurgence in the field. Over the past year researchers at Columbia's CRIS lab have made some important contributions in this area. Much of this work might be characterized as getting back to basics: we have revisited fundamental related work and examined the assumptions and validity of the models. The results - summarized here and detailed in other papers in these proceedings - may be a little surprising.

1.1 A Generalization of the Lambertian Model

The Lambertian assumption is one of the most widely used assumptions in machine vision. We have shown analytically as well as experimentally that rough surfaces, even when locally Lambertian, are non-Lambertian in reflectance. The paper [Oren and Nayar, 1993] details the development of a predictive model explaining these results. This work may shed light on an age old question: why the moon, which has a diffuse surface and is spherical, appears "flat."

Image brightness values are closely related to the reflectance properties of points in the scene. Hence, accurate reflectance models are fundamental to the advancement of machine vision. Recently, Nayar et al. [Nayar *et al.*, 1991] proposed a reflectance framework for machine vision that has three primary components: the diffuse lobe, the specular lobe, and the specular spike. The emphasis of their work was the analysis of the specular components rather than the diffuse component. The diffuse lobe was assumed to be *Lambertian* as this model is simple and does reasonably well in approximating reflection from a wide range of matte surfaces. A surface with Lambertian reflectance appears equally bright



(a) Real image

(b) Lambertian model

(c) Proposed model

Figure 1: Real image of a cylindrical clay vase compared with images rendered using the Lambertian reflectance model and the proposed diffuse reflectance model. The vase is illuminated by a point source from the viewing direction.

from all directions. This model for diffuse reflection is one of the most widely used assumptions in machine vision. It is used explicitly in the case of shape recovery techniques such as shape from shading and photometric stereo. It is also implicitly used in the solution of the correspondence problem by vision techniques such as stereo vision and motion detection.

For several real-world objects, however, the Lambertian model can prove to be a poor and inadequate approximation to the diffuse component. In the areas of machine vision, remote sensing, and computer graphics, each picture element (pixel) can represent a surface area with substantial roughness. Though the Lambertian assumption is often reasonable when looking at a small planar surface element, the roughness of the total surface covered by a pixel causes it to behave in a non-Lambertian manner. This deviation from Lambertian reflectance is significant for very rough surfaces, and increases with the angle of incidence. We have developed a comprehensive model that predicts reflectance from rough diffuse surfaces, and conducted several experiments that support the model [Oren and Nayar, 1993], [Oren and Nayar, 1992]. The proposed model takes into account complex geometrical effects such as *masking*, *shadowing*, and *interreflections* between points on the rough surface. It may be viewed as a generalization of the Lambertian reflectance model.

Figure 1(a) shows the image of a cylindrical clay vase with a rough surface. This image was obtained using a CCD camera. The vase is illuminated by a single light source from the sensor direction. Figure 1(b) shows a rendered image that is generated using the known geometry of the vase and the Lambertian model. Clearly, the real vase appears much flatter, with less brightness variation along its cross-section, than the Lambertian vase. Figure 1(c) shows a rendered image of the vase generated using the proposed reflectance model. Note that the proposed model does very well in predicting the appearance of the vase. Figure 2 compares the brightness values along the cross-section of the three vase images shown in Figure 1. It is interesting to note that the brightness of the real vase remains nearly constant over most of the cross-section and drops quickly to zero very close to the limbs. The proposed model does very well in predicting this behavior, while the Lambertian model produces large brightness errors. We note that in graphics rendering of diffuse objects, it is general practice (and an acknowledged hack) to add an "ambient light source" which modifies the reflectance function producing "flatter" renderings of objects and making them look less harsh. The proposed model can provide that desired behavior, and does so using rigorous foundations.

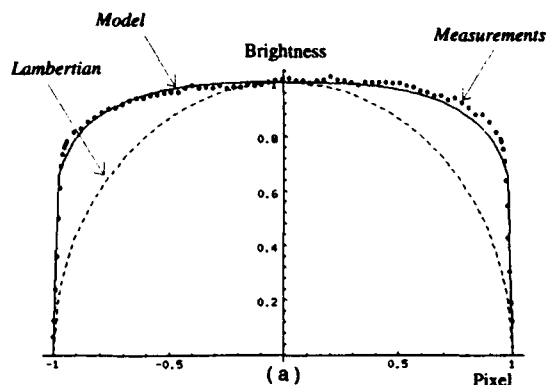


Figure 2: Comparison between the image brightness along the cross-section of three vases shown in Figure 1.

1.2 Polarization-Based Techniques

In the past three IUW proceedings, Columbia has presented reports discussing the benefits of polarization techniques in machine vision. Basic theory and numerous applications have been addressed. There have been, however, technical assumptions which may not have been apparent on first reading. Some of these assumptions turned out to be quite strong, and removing them proved more than a simple extension of the previous work.

1.2.1 Integration of Color and Polarization

Two serious assumptions were associated with the problem of separation of diffuse and specular components of an image. These assumptions restricted previous polarization-based work to dealing with compact (i.e. small) highlights over regions with constant diffuse components and constant material composition. Previous work on this problem using color information made similar restrictions. The separation problem is important since many image understanding algorithms, such as shape from shading, stereo, photometric stereo, and motion analysis, fail when there are significant highlights and specular interreflections.

The new approach uses color and polarization information to remove the restrictions of compact highlights and constant diffuse components. In Figure 3, we see a gray-scale version of a color image of a mug with a significant highlight, and after removal of that highlight. More details, as

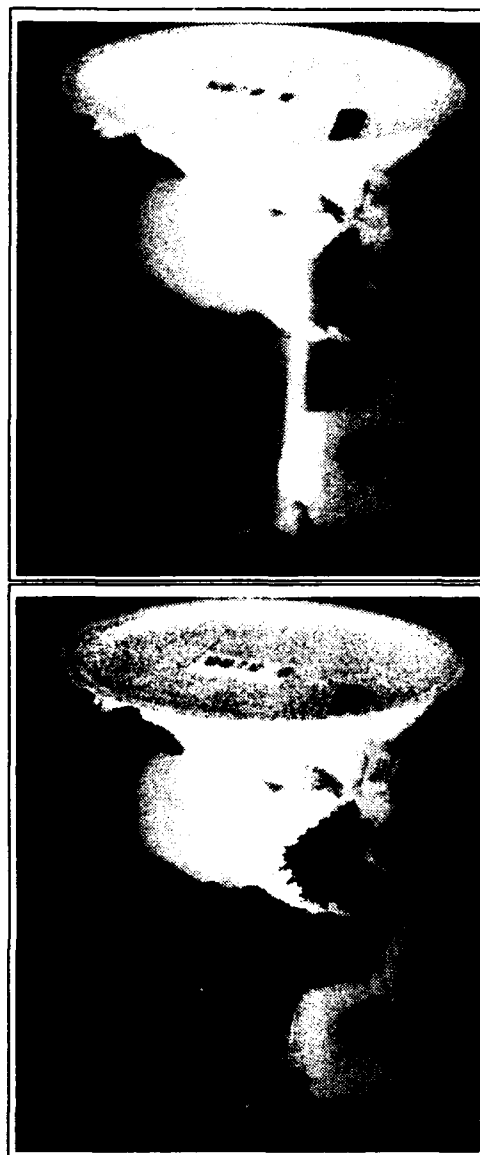


Figure 3: Top is the image of a cup with a strong highlight. Bottom is the diffuse component of the cup as computed with the integrated color/polarization approach. (These grayscale images were converted from color images, see [Nayar *et al.*, 1993] for color versions of this and other examples of separation.)

well as the color images from three experimental scenes, can be found in these proceedings [Nayar *et al.*, 1993].

The new approach is not just an application of previous work in three color bands with a simple combination of the results. Rather it started from the fundamental characterization of specular highlights/diffuse reflection and derived new

constraints in color space, which can be used to determine the diffuse reflection at a point.

We have tested the new technique in experiments using both a color camera and a monochrome camera with color filters, on scenes including a simple geometric shape (cylindrical cup), with strong surface markings and a strong primary highlight, see figure 3, and more complex scenes with strong interreflections, see [Nayar *et al.*, 1993]. The results of the experiments not only verify the approach as feasible, but show a considerable advance. The scenes considered include examples that would not be computable using previously existing algorithms. There are still a few assumptions to be overcome before the separation is widely useful: it does not handle near normal reflections or highlights that are nearly the same color as the object. Still, it represents an important advancement in the state of the art.

Unfortunately, the only available analysis of the separation quality is qualitative/subjective interpretation. Over the next year we will use this algorithm as a preprocessor to stereo, shape from shading and photometric stereo. Even with the aforementioned limitations, we expect that it will significantly help in these application areas. These methods recover depth or shape information and we will use the the resulting surface "quality metrics" as a quantitative measure of the impact of separation.

1.2.2 Polarization for the UGV: Stepping Outdoors

Other strong assumptions in our previous polarization work came from our exclusive use of indoor environments. In support of the UGV program we have teamed up with L. Wolff and Johns Hopkins University and L. Matthies at JPL to address the problem of material/scene classification in outdoor scenes. There are two things that make this quite challenging: dealing with the complexities of the outdoor world and collecting the necessary data.

The problems in outdoor scenes are many. First, natural skylight is partially polarized, and reflections of it are thus more complex. Second, outdoor scenes containing vegetation have complex shapes with significant internal structure

(consider the potential number of reflections between the leaves of a tree). Yet we generally view these from a distance at which most internal structure is within a pixel. As mentioned above, even for something as simple as Lambertian reflectance, complex sub-pixel surface geometries (i.e. roughness) can create formidable deviations from the model.



Figure 4: Image of a scene near JPL containing water, vegetation, bare soil, and a building.



Figure 5: Percent polarization the scene. [0% - 30%] \rightarrow [255 - 0]

The approach being explored by the joint effort is twofold: To venture forth and gather data for analysis, and to find ways to use additional information, e.g. vehicle pose and multi-spectral information. We have already gathered data for



Figure 6: Polarization phase image for the scene in figure 4. $[0^\circ - 180^\circ] \rightarrow [255 - 0]$

a number of scenes at 6 different wavelengths and are analyzing the data. Figures 4-6 show some of the polarization information obtained from the 650nm wavelength band for a scene near JPL. Shown are the visible image, the percent polarization and the polarization phase. The scene contains a body of water (in front) with a dirt/rock bank. Above the bank are various forms of vegetation, some bare soil, and in the background is a building. The percent polarization (figure 5) clearly delineates the water from the bank, but the reflections of some of the vegetation are considerably more complex. The direct view of the vegetation is also very detailed. The phase information has complexities rivaling those of the intensity image. We clearly have information, it is just not clear how to use it. The joint research effort is looking at how we can exploit known geometry (we can model some of the partial polarization effects of skylight, if we know the time, viewing direction, local weather conditions, and vehicle pose) and multi-spectral information.

A final problem in the use of polarization for the UGV project is that of data acquisition: how to get the polarization information on a moving vehicle. That is discussed in the section of the overview on real-time vision.

2 Object Recognition, Shape Recovery and Learning

Three key problems in high-level vision are Shape Recovery, Object Recognition and Learning of Object Models. This section overviews Columbia's recent and ongoing work in these key areas.

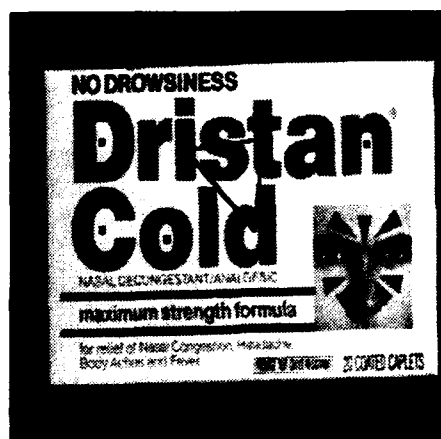
2.1 Reflectance-Based Recognition

Object recognition has been an active area of machine vision research for the past two decades. The traditional approach has been to recover geometric features from images and then use these features to hypothesize and verify the existence of three-dimensional objects in the image. Edges and vertices are examples of geometric features often used by recognition systems. During image formation, however, a substantial amount of information is lost regarding the geometry of the scene. Hence, geometric features are not always adequate for robust recognition of objects. In the past, little attention has been given to the use of other properties of objects for recognition. In addition to its geometry, an object may be characterized by physical properties such as reflectance, roughness, and material type.

An efficient algorithm has been developed for computing the reflectance of regions in a scene, with respect to their backgrounds, from a single image [Nayar and Bolle, 1993a] [Nayar and Bolle, 1993b]. The result is a physical property of each scene region that is invariant to the intensity and direction of illumination. This photometric invariant, referred to as the *reflectance ratio*, provides valuable information for recognition tasks. We have used the reflectance ratio invariant to recognize objects from a single image [Nayar and Bolle, 1993a]. This approach is very effective in the case of man-made objects that have printed characters and pictures. Each object is assumed to have a set of regions, each with constant reflectance. The reflectance ratio and center of each region are used to represent the object. Algorithms based on the indexing scheme have been developed for recognition and pose estimation of objects from a single image.

Experimental results are presented in [Nayar and Bolle, 1993a] for realistic scenes with occlusions, shadows, and illumination variations. Figure 7

shows object acquisition and recognition results obtained using the reflectance-based recognition algorithm. The image shown in Figure 7a is used for learning the object model. Three reflectance regions are used to form an index and are indicated by the triangle. Other regions on the object are included in the entry of a hash table to be used for object verification and pose estimation. The centroids of these verification regions are indicated by black boxes in Figure 7a. Figure 7b shows a scene with several objects. The index triangle is detected in the scene image and is shown as a triangle in Figure 7b. This provides a hypothesis for the object and its pose in the scene image. This hypothesis is verified by projecting other regions in the model image to the scene image.



(a)



(b)

Figure 7: Model acquisition and object recognition results obtained using the reflectance based recognition algorithm.

2.2 Object Recognition using BSP Trees

We recently began investigating the application of Binary Space Partitioning (BSP) trees to the domain of object recognition. A BSP tree is a general method of partitioning an n -dimensional space by a set of $n-1$ dimensional hyperplanes. Its tree structure allows very efficient algorithms to be developed, it is compact, and it is numerically robust. BSP trees have proven their utility in 3-D modeling, graphics and image processing, and many of the properties that allow BSP trees to perform well in these tasks are of primary importance when considering the object recognition problem.

The object recognition system we are developing uses augmented BSP trees, generated from CAD models, to model the objects being examined. Data is acquired from an image, a rangefinder, or tactile sensor, which the system then uses to find matching features in the BSP tree models. The tree structure of the model will allow the system to quickly obtain a correlation between the sensed and modeled features. Models will also be used to guide the sensing process towards features that are more discriminating over those that are less so. The output, which consists of the type of object, its position and orientation, may then be used to direct further sensor planning or tasks such as manipulation and path planning. In addition, a recognized object no longer needs to be associated with the sensed data and may be described by its model, which may be viewed as a compression of the sensed data.

2.3 Shape Recovery

In the past few years we have been developing and analyzing a range of different methods for shape recovery. Most of these projects continued this year, with incremental progress and a few publications with more experimentation or analysis. The basic concepts have been covered in past years. We report only the recent publications:

- Shape-From-Focus [Nayar, 1992a, Nayar, 1992b]
- Recovery of TORI from Range Images [Kjeldsen and Kender, 1992]

- Energy-Based Segmentation [Boult and Lerner, 1991], [Lerner, 1993].
- Shape from Shadows [Yang and Kender, 1993]
- PROVER [O'Donnell and Boult, 1991]
- Symmetry Analysis, SHGC Modeling and Recovery [Gross and Boult, 1992]

This year we have begun research using dynamic models combining previous work on splines, generalized cylinders and superquadrics.

2.4 Learning and Recognition of Objects from Appearance

A technique is being developed for automatically learning object models for recognition and pose estimation [Murase and Nayar, 1993]. In contrast to the traditional approach, we formulate the recognition problem as one of matching appearance rather than shape. The appearance of an object in a two-dimensional image depends on its shape, reflectance properties, pose in the scene, and the illumination conditions. While shape and reflectance are intrinsic properties of an object and are constant, pose and illumination vary from scene to scene. We have proposed a new compact representation of object appearance that is parameterized by pose and illumination. For each object of interest, a large set of images is obtained by automatically varying pose and illumination. This large image set is compressed to obtain a low-dimensional subspace, called the eigenspace, in which the object is represented as a hypersurface. Given an unknown input image, the recognition system projects the image onto the eigenspace. The object is recognized based on the hypersurface it lies on. The exact position of the projection on the hypersurface determines the object's pose in the image.

Experiments have been conducted using several objects with complex appearance characteristics [Murase and Nayar, 1993], [Murase and Nayar, 1992]. Fig. 8a shows an input image of the object (car) whose parametric hypersurface representation is shown in Fig. 8b. The representation is parameterized by object pose (θ_1) and illumination direction (θ_2). The hypersurface is actually 8-dimensional but only the three most prominent dimensions are displayed in Fig. 8b. The input image in Fig. 8a is projected onto

the eigenspace and is seen to lie on the parametric hypersurface of the object. The location of the point on the hypersurface determines the object's pose in the image. The performance of the recognition and pose estimation algorithms is studied using over a thousand input images of the sample objects [Murase and Nayar, 1992]. The sensitivity of recognition to the number of eigenspace dimensions, and the number of learning samples, is analyzed. For the objects used, appearance representation in eigenspaces with less than 10 dimensions produces very accurate recognition results with an average pose estimation error of 0.5 degrees. These results suggest the proposed appearance representation to be a valuable tool for a variety of machine vision applications.

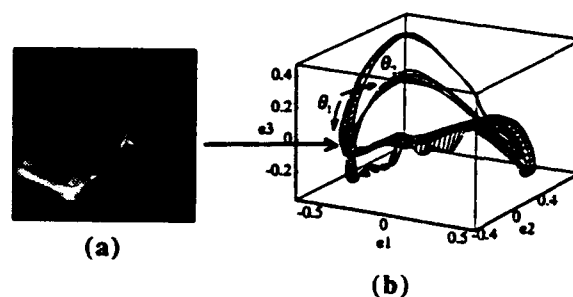


Figure 8: (a) An input image. (b) The input image is mapped to a point in eigenspace. The location of the point on the parametric hypersurface representation of the object determines its pose in the input image.

3 Sensor and Illumination Planning/Modeling

This section overviews two important aspects of using sensors. The first is the planning of parameters for sensors and illuminators, e.g., illumination placement, sensor placement, sensor lens settings, etc. The second area is the modeling of sensors and uncertainty in robotic/vision systems. These research projects go to the heart of a key problem in vision—how we can reason about IU systems and *plan* to have them work rather than getting them to work via fiddling around with the sensor positions, light position, etc.

3.1 Sensor Planning in Active Environments

A goal of robotics has been to develop intelligent robots which are capable of planning their own actions. These actions are often guided by sensors, which provide noisy, incomplete, and sometimes inaccurate data. Researchers accept this as a necessary evil of sensors and develop algorithms which try to extract as much information as possible from sensor data. However, less attention has been focused on planning sensing strategies which yield more accurate data. The Machine Vision Planning (MVP) system developed at Columbia University attacks this problem by integrating sensor, object, and motion models, along with task-level information to plan appropriate sensor locations and settings, see [Abrams *et al.*, 1993a, Timcenko *et al.*, 1993, Abrams *et al.*, 1993b].

Given a CAD description of an object and its environment, a model of a vision sensor, plus a specification of the features to be viewed, MVP generates a camera location, orientation, and lens settings (focus-ring adjustment, focal length, aperture) which insure a robust view of object features. In this context, a robust view implies a view which is unobstructed, in focus, properly magnified, and well-centered within the field-of-view. In addition, MVP attempts to find a viewpoint with as much margin for error in all parameters as possible.

We have added moving environment models to MVP and are exploring methods of extending MVP to plan viewpoints in an active environment. The first approach, currently limited to the case of moving obstacles (i.e. the target, or features to be viewed, are stationary), is to sweep the model of all moving objects along their trajectories and to plan around the swept volumes, as opposed to the actual objects. A temporal interval search is used in conjunction with the swept volumes to find large time intervals for which one robust viewpoint can be used. This approach has been implemented in simulation and experiments are being carried out in our robotics laboratory.

The lab setup involves two robot arms. The first arm has a camera mounted on its end-effector that can be positioned anywhere in the environ-

ment in order to monitor a robotic task. The second arm is given a particular task such as a pick-and-place or welding operation. The goal is to have the MVP system plan viewpoints which allow us to monitor the task, while avoiding occlusion which may arise due to the motion of the second robot. Future research will concentrate on planning continuous paths that link the viewpoints together, and continuing to create a fully automated environment for active sensing tasks [Timcenko *et al.*, 1993].

3.2 Illumination Planning

Using the parametric eigenspace representation (described in above section on learning) we have developed a new approach to the problem of illumination planning, see [Murase and Nayar, 1993]. Given a set of objects, the goal is to determine the illumination direction for which the objects are most distinguishable in appearance from each other. Correlation is used as a measure of the similarity in the appearance of objects. For each object, a large number of images are automatically obtained by varying pose and illumination direction. The sets of images for each object constitutes the planning set. A parametric eigenspace is computed using the planning image set. For each illumination direction, objects are represented as hypercurves in the eigenspace. The minimum distance between the hypercurves of two objects represents the similarity between the objects in the correlation sense. The optimal source direction is one that maximizes the shortest distance between object hypercurves in eigenspace. This illumination planning method has the following advantages over previous approaches:

- (a) It does not rely on geometric (CAD) models of the objects of interest. It uses only brightness images to accomplish the planning task.
- (b) No assumptions are made with respect to the reflectance properties of the objects.
- (c) The optimal source direction produced by the illumination planner is pose invariant; it is optimal over all object poses.

3.3 Modeling Uncertainty

In [Timcenko and Allen, 1992, Timcenko and Allen, 1993b] we offer a new method for modeling uncertainties that exist in a robotic system.

based on stochastic differential equations. The benefit of using such a model is that we are then able to capture in an analytical structure some key points underlying robot motion: the ability to properly express uncertainty within the motion descriptions, and the dynamic, changing nature of the task and its constraints. The goal of this research is to exploit the smooth, differentiable topological structure of configuration space and populate it with mathematical entities that lead to plans as solutions of certain differential equations.

We have performed experiments that attempt to quantify the uncertainty in robotic motion control and show how it can be used within our model. The statistical justifiability of the proposed model indicates that it resembles the real nature of the random phenomena that govern the system quite well. More importantly, the method offers a way of estimating the variance of different types of uncertainties, thus answering questions about both the qualitative and quantitative nature of uncertainty.

Also of interest is defining methods for time parameterization of robot trajectories so that the robotic system achieves favorable performance *despite* the inevitable presence of uncertainties [Timcenko and Allen, 1993a]. In order to develop these methods, we need detailed understanding of the sources of random phenomena in the system, as well as comprehensive mathematical models of those phenomena. That understanding can lead us to a mathematically tractable formulation of motion planning in the form of a constrained optimization problem.

We believe that this is a fruitful research direction. It opens a wide spectrum of questions. Some of them are: 1) dealing with non-constant environment uncertainty, 2) the robustness of obtained plans with respect to modeling errors, 3) the numerical complexity of computing approximations of globally optimal plans, 4) experimentation with different types of difficulty indices and different types of constraint functions, such as mathematical expectations instead of success probabilities. We hope to address some of these problems in future.

3.4 Sensors and the IUE

The DARPA Image Understanding Environments specification was recently released. (See also the papers on the IUE in this proceedings and also [Mundy *et al.*, 1992]. Unofficial copies of both the IUE Overview Document and the IUE Class Definition Document, can be FTPed from cs.columbia.edu in the directory /pub/vision/iue.) A major part of this document was the specification of sensors and related objects. Our group was instrumental in this part of the specification, drawing partially from our experiences in the PROVER project. The design incorporates both sensor data characteristics and sensor uncertainty modeling. This past year we have been prototyping some of the IUE sensor objects, providing feedback on the design. Our experiences should be summarized in a technical report later this year.

4 Qualitative Vision

In Columbia's CRIS lab we have initiated projects in the area of qualitative vision - projects where the goals of vision are not measurements but qualitatively describe goals such as object alignment without calibration, navigation with topological directions or determine the applicability of certain propositions (such as near, far, next to) to describe the relationship of objects in a scene. In this section, we review our recent work in these areas. It is interesting to note that while the title of the section is "qualitative vision," all the projects have a significant quantitative component. For example, if we did topological navigation without a quantitative error analysis, it would be difficult for us to measure research progress.

4.1 Alignment using an Uncalibrated Camera System

This new technique takes the typical mapping from 3-D positions to image coordinates, and instead of finding this mapping, it recovers a property of the image coordinates without calibrating the camera location. This method exploits the fact that a known movement in the camera system can result in useful motion information in the image system without knowing the exact calibration between the systems.

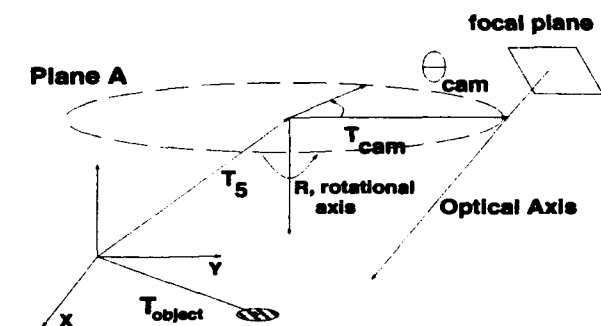
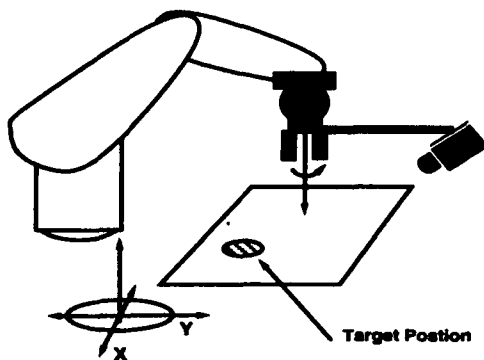


Figure 9: Experimental setup for uncalibrated alignment



Figure 10: Work table with "U" shaped part to be tracked

In this method, an active camera system mounted on a robotic arm can maintain an arbitrary, geometric relationship with an object system, and as a result of certain operations, the vision system can "calibrate" itself to or "can define its location with respect to" the unknown camera system. The newness of our technique arises from the fact that our system "calibrates"

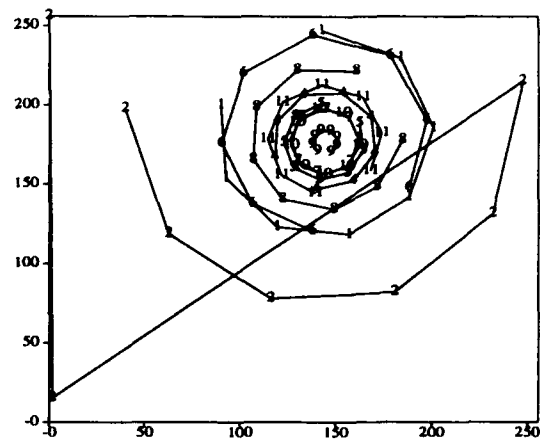


Figure 11: The image-space projections of the ellipses formed by tracking the object over the first 11 positions moved to by the robot.

itself while performing the task of moving to the goal position and without computing the true location of the camera system.

Given an alignment task (e.g. insertion of a part in an assembly), we first select designated features on the part to be inserted. The robot then rotates the camera around its rotational axis, R . If the only movement in the robot-camera system is the rotation, the part features will trace out a conic section, an ellipse under certain conditions. By noting the changes in these elliptical parameters as the camera system moves (and computing the ellipse's projected area), we can recover the alignment condition. We determine the object's position based on the fact that if an object lies on the axis around which a camera system moves, the object will rotate but will not translate in the camera system. This fact allows us to set up a control structure for closed-loop servoing to the alignment position even though we have no knowledge of the camera calibration information, see [Yoshimi and Allen, 1993].

Figure 9 shows the setup of the camera mounted on the arm. Figure 10 shows a workspace with a part to be tracked for alignment; the goal is to insert a pin in the hole of "U" shaped flange. Figure 11 shows the generated ellipses from tracking the hole during the camera's rotation, and the convergence of the ellipse data as the alignment succeeds.

4.2 Topological Navigation

Navigation in a unstructured environment is generally carried out in a topological manner

with directions like “turn right when you see the isolated yellow house, take a left at the second light,” etc.) rather than using metric directions like “travel 1.345 miles, turn right, go 3.12 miles turn left”. This past year we have continued our work in this area, working on an analytic (quantitative) model of the error behavior of our system. In [Park and Kender, 1992] we describe a purely topological method for navigation in a large unstructured environment that contains featureless objects, using qualitative non-metric information such as “isolated” landmarks and “trajectories,” which we define. The map-maker and the navigator are implemented using an IBM 7575 SCARA robot arm, PIPE, and two cameras. The navigational environment consists of a flat plane with identical objects populated randomly but densely on it. First, given a starting position and a goal position the map-maker module observes the environment and generates a “custom map” that describes, in a non-metric language, how to get from the starting position to the goal position efficiently and reliably. The accuracy and the cost of the directional instructions are analyzed, then demonstrated by the navigator by following the commands in the custom map.

The errors in denoting isolated objects as landmarks were analytically determined, and combined with an analytic determination of errors in using landmark pairs to denote parkway-crossing directions. These two error measures are then treated as a single measure of topological goodness. Since both error measures are parameterized by a model of sensor error (the standard deviation of landmark exact placement), it was illuminating to simulate the variations in paths produced under increasing error, particularly in cluttered environments. As error increases, crowded areas are avoided. For more details see [Park and Kender, 1993] in these proceedings.

4.3 Qualitative Spatial Description

In document understanding, the relation between image understanding and language becomes very important. We have been looking at the relationship between objects within an image and the automatic determination of prepositions to describe them, see [Abella and Kender, 1993]. In the area of spatial descriptions, the problem of

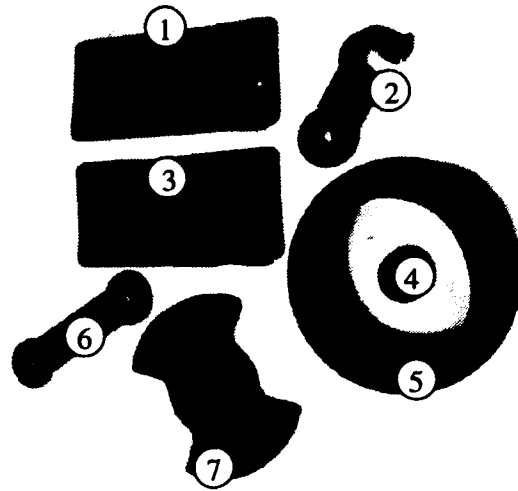


Figure 12: An image for experimentation with recovery of spatial descriptions. See text for details.

using prepositions of place, such as “near,” “in,” or “next to,” was formalized in a translation-independent, rotation-independent, and scale-independent way. Using prior research on the descriptions of quadrilaterals, the method allows fuzzy, probabilistic estimates on how accurately a given preposition describes the two-dimensional relationship of objects.

We will use the image shown in figure 12 to illustrate several “recovered” prepositions. Each object has been numbered to ease its reference. Internally, objects in the system are represented using moments through second order. Currently the system accepts a spatial preposition and displays all those objects that satisfy the preposition inequalities. The system also accepts as input two objects along with a preposition and it outputs how well those two objects meet the given preposition (the value of f_{U_p} for given σ). All intuitively obvious relations between objects are discovered by the system, e.g. objects 1 and 3 are *next to* each other, 4 is *in* 5, 6 *next to* 7, etc.

An interesting case, and one that demonstrates the effects of fuzzification is the case of supplying object 2 and object 6 along with the preposition *aligned*. Fuzzification is accomplished by blurring the object with a Gaussian distribution with a standard deviation of σ . With no fuzzification the system finds that 2 and 6 are not aligned. However, if we allow a certain amount of fuzzifi-

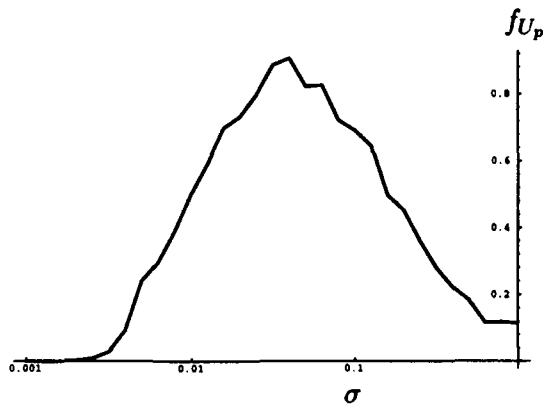


Figure 13: The dependency of $f_{U_{aligned}}(2,6)$ as a function of σ

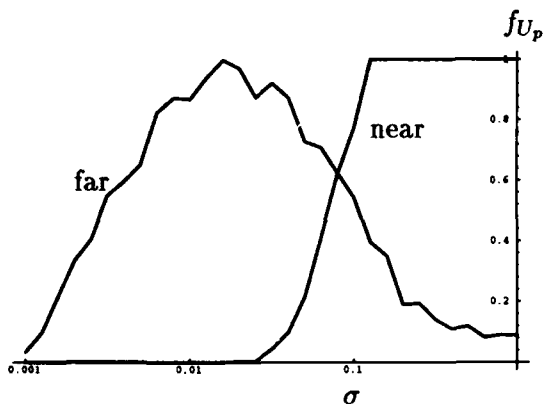


Figure 14: The dependency of $f_{U_{near}}(2,6)$ and $f_{U_{far}}(2,6)$ as a function of σ

cation with say $\sigma = 0.03$ the value of $f_{U_{aligned}}$ is 0.8. This value indicates that they may be sufficiently aligned to be regarded as such (which we actually see in the image!), depending on how much leeway we wish to allow. The dependency of $f_{U_{aligned}}$ on σ is shown in figure 13. From this graph we see that the value of the membership function significantly deteriorates for large values of σ . This simply means that the amount of induced uncertainty is so large that the objects cease to possess their original features (such as orientation in this case). This also indicates what the maximal acceptable value for σ should be, in this case, $\sigma < 0.1$.

Another interesting case is that of supplying object 2 and object 6 along with the preposition *near* or *far*. Neither satisfies the inequalities precisely. However, if we again, allow for fuzzification, we get a most interesting result, as shown in figure 14. We observe that although we can

not say for certain that object 2 and object 6 are either *near* or *far*, we can say that they are *somewhat near* or *somewhat far*. How we decide which of the two to use can be seen in figure 14. If we examine the slopes of the two curves we see that for small values of σ the slope for *far* is steeper than that for *near*. Therefore it would seem more appropriate to say that 2 is *somewhat far* from 6 as opposed to 2 is *somewhat near* to 6.

5 Real-Time Vision

While not really a "vision topic" in itself, doing vision in real-time often means crafting clever algorithms to fit the hardware and timing constraints. The topics in this section are all related to some research topic above, but are separated out because of their real-time nature.

5.1 Real-Time Polarization Computation

A problem in the use of polarization for the UGV project is that of data acquisition: how to get the polarization information on a moving vehicle. Before we can develop significant real-time polarization computation algorithms, we need the data. Previous algorithms at Columbia used a rotating filter in front of a single camera. While sufficient for indoor inspection-type tasks, it is not acceptable for vehicles. In the past year we have had two projects addressing the real-time polarization computation problem: one based on beam-splitting, and one on image warping. We have already implemented and tested the algorithms for polarization computation; it is just a question of getting multiple images, registered in space and time, which measure different polarization states.

5.1.1 Polarization Preserving Beam-splitter (PPBS)

We have designed and built a "polarization preserving beam splitter," and are in the process of calibrating it. (Actually, it does not totally preserve polarization, but the effects introduced can be compensated for via calibration.) We have found that with its 20 degree field of view and beam-splitting external to the camera optics, precise pixel alignment is difficult to achieve,

and, more importantly, virtually impossible to maintain. We were also not able to achieve it simultaneously over the entire image in different spectral bands. Our conclusion was that we would work with this approximate alignment, calibrate the different distortions, and then warp the images to achieve alignment. This is similar to the approach we used last year to deal with chromatic aberration correction in regular lenses [Boult and Wolberg, 1992a]. The calibration process is now being implemented. The PPBS is intended to allow internal experimentation and will be used as a comparison for real-time warping-based polarization computations. The PPBS is not suitable for use on the UGV vehicle.

5.1.2 Polarization Computation by Image-Warping

A second approach for real-time polarization, currently under investigation, is the use of real-time image warping to align images taken with separate cameras and then apply the computations. For small baselines (1-2 inches) and small focal length lenses (6-8mm) the disparity-distortions are rather small. The real issue is how accurately we can warp the image data, which is addressed in the next section.

5.2 Real-Time Image Warping

Our previous work in chromatic aberration correction ([Boult and Wolberg, 1992a, Boult and Wolberg, 1992b]) and our ongoing work in real-time polarization algorithm development, have demonstrated the need for high-quality image warping. While we have developed a serial algorithm for this, which requires 1-2 seconds on a standard workstation, the real-time polarization work requires real-time warping. The lens correction would also be more easily used if it were run in real-time as part of the image acquisition process. In the past year we have been developing and testing a near real-time, high-quality image warping algorithm. The algorithm was designed for the PIPE image processor, and currently runs at 10hz. Multiple versions can be run in parallel to allow 30hz processing with a 1/10 second pipeline delay. It uses separable image reconstruction filters (up to 9x9) and pre-computed distortion tables. We are currently testing it using the image reconstruction filters

developed under our previous DARPA contract, see [Boult and Wolberg, 1993]. In addition, we are developing the calibration processes to determine the spatial warp needed for different applications. Related work is also under investigation, with no significant results yet, on flexible multi-modal registration which uses explicit sensor models and a flexible matching process.

5.3 Real-Time Tracking

An important use of real-time vision is for tracking moving objects. In this area we have two projects, one on tracking objects for grasping by an arm, and the other on tracking for surveillance.

5.3.1 Real-Time Tracking and Grasping

Research in real-time motion tracking and grasping has succeeded in laboratory demonstrations of tracking and grasping moving objects. The focus of this work is to achieve a high-level of interaction between a real-time vision system capable of tracking moving objects in 3-D and a robot arm equipped with a dextrous hand that can be used to pick up a moving object. The system we have built addresses three distinct problems in robotic hand-eye coordination for grasping moving objects: fast computation of 3-D motion parameters from vision, predictive control of a moving robotic arm to track a moving object, and grasp planning. The system is able to operate at approximately human arm movement rates, and has been tested with a moving model train which is tracked, stably grasped, and picked up by the system (see Figure 15). The system uses a real-time motion stereo computation and a novel probabilistic filter [Allen *et al.*, 1992, Allen *et al.*, in press]. We have also published our results on unifying image-flow computations in an estimation-theoretic framework [Singh and Allen, 1992].

5.3.2 Surveillance Mode Tracking

Related ongoing real-time tracking work for surveillance is joint work with, and partially funded by, Texas Instruments. The approach builds upon some of Columbia's recent work in real-time tracking, but uses the DATACUBE MV20 processor as opposed to the PIPE. This

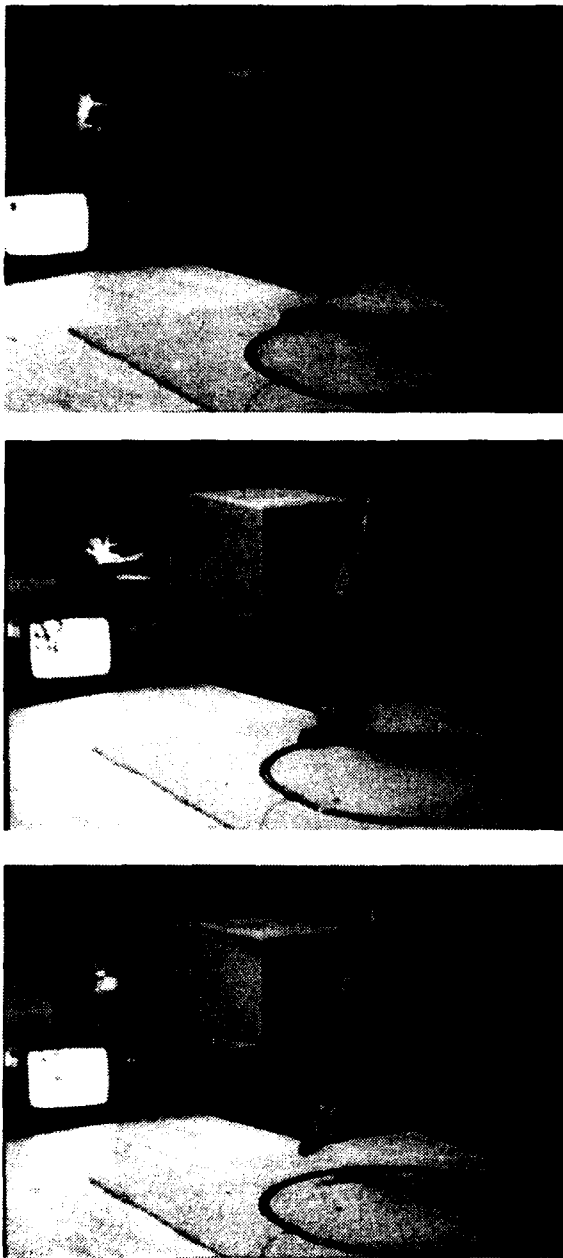


Figure 15: Intercepting, grasping and picking up the object

change in real-time hardware, plus the panning action of the camera, necessitated a moderately different approach. We are currently expanding the approach to be applied with a second generation FLIR, and expect to incorporate both sensor models and "target" motion models. This year's project will provide new information which should be useful in Columbia's own tracking research, as well as input for our work on Sensor Modeling. This technology transfer is thus bi-directional.

5.4 Real-Time Detail-Preserving Smoothing

This past year we also completed a formal description of our ongoing work on G-neighbor-based processing, see [Boult *et al.*, 1992]. G-neighbor algorithms use a modification of the usual 8-connected neighborhood, including only pixels that differ by less than a fixed amount or by less than a fixed ratio. These signal dependent neighborhoods are then used in traditional processing for detail-preserving smoothing or signal-dependent morphology. In [Boult *et al.*, 1992] we present a qualitative analysis of the usefulness of G-neighbor-based detail-preserving smoothing, comparing it to four previously existing algorithms for detail-preserving smoothing. The G-neighbor approach is significantly cheaper and easily parallelized. The results of the qualitative comparison were surprisingly good: the new algorithm was as good or better than the comparison algorithms. The quantitative analysis of these algorithms is still ongoing, and should be completed in the next year.

The algorithm has been implemented both on a workstation, under KBVision, and on our PIPE. The real-time system computes G-neighborhoods in a single frame-time and performs 1 iteration of smoothing per frame. For comparison, an example of the algorithm and a few comparison algorithms can be found in figure 16.

References

- [Abella and Kender, 1993] A. Abella and J. R. Kender. Qualitatively describing objects using spatial prepositions. Submitted. Available as a technical report.
- [Abrams *et al.*, 1993a] Steven Abrams, Peter K. Allen and Konstantinos A. Tarabanis. Dynamic sensor planning. In *International Conference on Intelligent Autonomous Systems*, Pittsburgh, PA, February 1993.
- [Abrams *et al.*, 1993b] Steven Abrams, P.K. Allen and K. Tarabanis. Dynamic sensor planning. In *Proc. DARPA Image Understanding Workshop*, Washington, 1993. These proceedings.
- [Allen *et al.*, 1992] P. K. Allen, A Alex Timcenko, B. Yoshimi, and P. Michelman. Trajectory filtering and prediction for automated tracking



Figure 16: Detail-preserving algorithms applied to a simple image with uniform noise. The upper right is the original image of a line-drawing, with a dynamic range of 38. The upper left shows it with added uniform noise (range ± 5). The middle left shows the result of 1 iteration G-neighbor smoothing with a G of 10. The lower left is the result of 1 iteration of Weymouth's algorithm with parameters .3 .3. The lower right is the result of using a Nagao operation with a 3×3 window. The middle right is the result of using 3×3 median filtering, with the central pixel given 5 votes. (If we use median with uniform votes, the detail is almost totally lost!)

and grasping of a moving object. In *IEEE International Conference on Robotics & Automation*, pages 1850 – 1856, Nice, 1992.

[Allen *et al.*, in press] Peter Allen, Aleksandar Timcenko, Billibon Yoshimi, and Paul Michelman. Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Transactions on Robotics and Automation*, [in press].

[Boult and Lerner, 1991] T.E. Boult and M.A. Lerner. Energy-based segmentation of sparse range data. In *Curves and Surfaces*, pages 43–50. Academic Press, NYC NY, 1991.

[Boult and Wolberg, 1992a] T.E. Boult and G. Wolberg. Correcting chromatic aberrations using image warping. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 684–687, Urbana, IL, June 1992.

[Boult and Wolberg, 1992b] T.E. Boult and G. Wolberg. Correcting chromatic aberrations using image warping. In *Proceedings of the DARPA Image Understanding Workshop*, pages 363–378, San Diego, January 1992. DARPA.

[Boult and Wolberg, 1993] T.E. Boult and G. Wolberg. Local image reconstruction and sub-pixel restoration algorithms. *Computer graphics and image processing: Graphical models and Image processing (CVGIP:GMIP)*, 1993. To appear. Available via anonymous ftp from cs.columbia.edu in directory /pub/vision/boult.

[Boult *et al.*, 1992] T.E. Boult, B. Melter, F. Skorina, and I. Stojmenovic. G-neighbors. Technical report, Columbia Univ. Center for Research in Intelligent Systems, November 1992. Submitted.

[Gross and Boult, 1992] A.D. Gross and T.E. Boult. Analyzing skewed symmetries. Technical report, Columbia Univ. Center for Research in Intelligent Systems, 1992. Submitted for pub, in revision.

[Kjeldsen and Kender, 1992] R. Kjeldsen and J.R. Kender. On seeing spaghetti: Self-adjusting piece-wise toroidal recognition of flexible extruded objects. Technical report, Columbia Univ. Center for Research in Intelligent Systems, 1992. Submitted for pub.

[Lerner, 1993] M.A. Lerner. *Energy-based segmentation*. PhD thesis, Columbia University Department of Computer Science, 1993.

[Mundy *et al.*, 1992] J. Mundy, T. Binford, T. Boult, A. Hanson, R. Beveridge, R. Haralick, V. Ramesh, C. Kohl, D. Lawton, D. Morgan, and

- K. Price. The image understanding environments program. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 406-417, June 1992.
- [Murase and Nayar, 1992] H. Murase and S.K. Nayar. Parametric eigenspace representation for visual learning and recognition. Technical Report CUCS-054-92, Columbia Univ. Center for Research in Intelligent Systems, 1992.
- [Murase and Nayar, 1993] H. Murase and S.K. Nayar. Visual learning of object models from appearance. In *Proceedings of the DARPA Image Understanding Workshop*, Washington D.C, April 1993. DARPA. These proceedings.
- [Nayar and Bolle, 1993a] S.K. Nayar and R.M. Bolle. Reflectance based recognition. In *Proceedings of the DARPA Image Understanding Workshop*, Washington D.C, April 1993. DARPA. These proceedings.
- [Nayar and Bolle, 1993b] S.K. Nayar and R.M. Bolle. Reflectance ratio: A photometric invariant for object recognition. In *Proceedings of the IEEE Computer Society International Conference on Computer Vision*, Berlin, 1993. IEEE. To appear.
- [Nayar et al., 1991] S.K. Nayar, K. Ikeuchi and T. Kanade. Surface reflection: Physical and geometrical perspectives. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-13(7):611-634, July 1991.
- [Nayar et al., 1993] S.K. Nayar, X.S. Fang and T.E. Boult. Separation of reflection components using color and polarization. In *Proceedings of the DARPA Image Understanding Workshop*, Washington D.C, April 1993. DARPA. These proceedings.
- [Nayar, 1992a] S. K. Nayar. Shape from focus system. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Urbana, IL, June 1992.
- [Nayar, 1992b] S. K. Nayar. Shape recovery methods for visual inspection. In *Proceedings of IEEE Workshop on Vision Applications*, Palm Springs, CA, December 1992.
- [O'Donnell and Boult, 1991] T. O'Donnell and T.E. Boult. Introduction of explicit sensor models in parametric object recovery. In *Proceedings of the SPIE Conference Sensor Fusion*. SPIE, November 1991.
- [Oren and Nayar, 1992] M. Oren and S.K. Nayar. Diffuse reflectance model for rough surfaces. Technical Report CUCS-057-92, Columbia Univ. Center for Research in Intelligent Systems, November 1992.
- [Oren and Nayar, 1993] M. Oren and S.K. Nayar. Generalization of the lambertian model. In *Proceedings of the DARPA Image Understanding Workshop*, Washington D.C, April 1993. DARPA. These proceedings.
- [Park and Kender, 1992] Il-Pyung Park and John R. Kender. Qualitative navigation using isolated landmarks. In *SPIE International Symposia*, 1992.
- [Park and Kender, 1993] I.P. Park and J. R. Kender. Qualitative environmental navigation. In *Proceedings of the DARPA Image Understanding Workshop*, Washington D.C, April 1993. DARPA. These proceedings.
- [Singh and Allen, 1992] Ajit Singh and Peter K. Allen. Image-flow computation: An estimation-theoretic framework and a unified perspective. *CVGIP: Image Understanding*, 56(2):152-177, September 1992.
- [Timcenko and Allen, 1992] A. Timcenko and P. Allen. Modeling uncertainties in robot motions. In *Applications of AI to Real-World Autonomous Mobile Robots, AAAI Fall Symposium Series*, 1992.
- [Timcenko and Allen, 1993a] A. Timcenko and P. Allen. "continuous model of uncertainties in robot motions based on stochastic differential equations". In *Proc. IEEE International Conf. on Robotics and Automation*, 1993. To appear.
- [Timcenko and Allen, 1993b] A. Timcenko and P. Allen. "peg insertion planning based on continuous uncertainty model". In *Proc. IEEE International Conf. on Robotics and Automation*, 1993. To appear.
- [Timcenko et al., 1993] A. Timcenko, S. Abrams and P. K. Allen. APHRODITE: Intelligent planning, control and sensing in a distributed robotic system. In *International Conference on Intelligent Autonomous Systems*, Pittsburgh, PA, February 1993.
- [Yang and Kender, 1993] D. Yang and J.R. Kender. Shape from shadows in the presence of data noise. In *Proceedings of the DARPA Image Understanding Workshop*, Washington D.C, April 1993. DARPA. These proceedings.
- [Yoshimi and Allen, 1993] Billibon Yoshimi and P. K. Allen. Alignment using an uncalibrated camera system. In *Proc. DARPA Image Understanding Workshop*, Washington, 1993. These proceedings.

USC IMAGE UNDERSTANDING RESEARCH: 1992-1993

R. Nevatia, K. Price and G. Medioni*

Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273

Abstract

This paper summarizes the USC Image Understanding research projects and provides references to more detailed sources of information. Our work has focused on the topics of 3-D vision (including range data processing, stereo, shape from contour and object recognition), aerial image analysis, motion analysis (including 3-D motion and structure estimation, visual guidance for mobile robots, and an integrated motion system), and parallel processing (including mapping algorithms onto specific or flexible architectures, and processor-time trade-offs).

1 INTRODUCTION

This paper summarizes our research projects over the last year. Much of this work is described in detail in the other papers in this proceedings, with this overview giving only a brief description of the detailed efforts. Work that is less complete will be described in somewhat more detail in this overview since there is no corresponding paper in the proceedings.

Our research covers a broad range of separate tasks in image understanding, but the different tasks are highly inter-related and share many common techniques. The four major task areas are three-dimensional vision, motion analysis, parallel processing and aerial image analysis, which is largely supported under the RADIUS program. This introduction will briefly describe the different research projects.

Three-Dimensional Vision

Three-dimensional vision is needed for many tasks including those of manufacturing, robotics and outdoor object recognition. To achieve these goals, we must develop a representation formalism that is rich enough to

describe complex objects in terms of both volumes and surfaces and which allows us to convert between them. We must also develop techniques to compute these descriptions from real data which includes shadows and noise and we must recognize such objects from a large database of objects. Three-dimensional vision is the largest research area funded under this contract. Within that area, we have a variety of projects:

- *Description of 3-D objects:* We are studying the problem of generating 3-D surface descriptions from range data using the concept of deformable models. This work is described in [Liao & Medioni 1993]. A second project using range data is exploring the integration of surface descriptions from different viewpoints and is discussed in detail in [Chen & Medioni 1993]. A third effort addresses the problem of recovering segmented, hierarchical volumetric descriptions from range data.
- *Perceptual Grouping:* Most high level vision algorithms require perfect data as input, but it is impossible to generate such features with low level algorithms such as edge detectors. We are working on bridging this gap by transforming an edge image into a saliency map. This approach uses a non-iterative method based on a field associated with each edge. This field encodes the notions of simplicity, curvature constancy and co-curvilinearity. A detailed report on this effort is given in [Guy & Medioni 1993].
- *3-D Shape from Monocular Images:* In this project, we are developing techniques for inferring 3-d shape descriptions given only a single image of the scene. We have developed techniques to use our earlier theoretical results on real images where contours are likely to be fragmented and distracting contours such as markings and shadows are present. We report on these efforts in [Zerroug & Nevatia 1993a and b].

Motion Analysis

We have a number of projects in the area of motion analysis, with autonomous navigation providing the context

* This research was supported in part by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-90-C-0078. Some work was supported under the RADIUS program under a sub-contract from Hughes Aircraft Co. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

for most of the work, though these techniques have a much broader utility.

- *Integrated system for Motion:* We have developed an integrated system that includes hierarchical feature extraction and matching and feedback of 3-D motion estimation to the feature matching process. The system is able to tolerate errors and differences in the feature extraction and matching process by removing these inconsistent feature points from the later analysis. This is described in [Kim & Price 1993].

- *Mobile Platform:* We use the domain of autonomous navigation to unify our motion work. To this end we have a small project in vision based navigation with a trinocular stereo system for reliable 3-D descriptions of the environment. The recent results for this effort are briefly given later in Section 3. More details can be found in [Kim & Nevatia 1993].

Aerial Image Analysis

Our work in aerial image analysis consists of two major components. First is the transfer of technology funded by DARPA to the RADIUS program. We also continue on our long range effort of analyzing complex cultural domains. Our recent work in extraction of buildings is given in [Huertas *et al.* 1993] and [Chung & Nevatia 1992b]. In the domain of large commercial airports we have shown good results on the detection of runways and taxiways [Huertas *et al.* 1990]. Recently we have ported this system from the older Symbolics version to the Sun platform and assisted another DARPA funded group at USC in the reimplementation of these algorithms on a Prolog machine. This airport analysis work also forms the basis for a new effort in using the Loom knowledge representation system for vision tasks.

Knowledge Based Vision

We have begun a project in using a standard knowledge representation system (the Loom system developed at USC-ISI). This system will be applied to our earlier airport analysis system. This work is briefly described in Section 5.

Parallel Processing

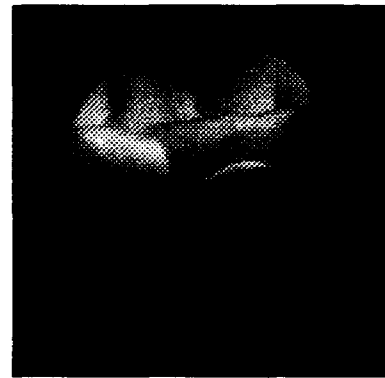
We are investigating parallel implementations of various vision algorithms developed in our group and elsewhere. We have studied algorithms for stereo and image matching, graph algorithms. This involves implementing such algorithms on existing architectures. The recent work on implementing scalable data parallel geometric hashing form matching is discussed in more detail in [Khokhar & Prasanna 1993]. In previous work, we have also explored the advantages of using flexible architectures [Reinhart 1991].

2 THREE-DIMENSIONAL VISION

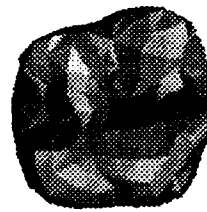
2.1 Description of 3-D Objects

2.1.1 Integration from Multiple Views

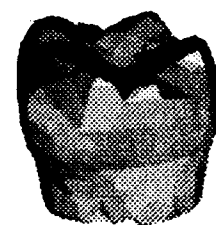
We have developed systems for building models from unregistered multiple range images [Parvin & Medioni 1992, Chen & Medioni 1992]. The latter system integrates views at the triangulated surface level rather than at the pixel level. A triangulated surface model can represent a variety of solid objects, and theoretically to any kind of resolution. They are not ideal representations for high level vision tasks, such as recognition, because, first, the representation is still low level, second, it is sensitive to many parameters, and therefore unstable. However, we think it is a good intermediate representation for integration and for building high level description through surface interpolation from triangulation. Figure 1 shows the multi-view integration result for a complex object.



(a) Original intensity image



(b) Rendered result



(c) Rendered result

Figure 1 Three dimensional reconstruction from several views. Two views of the resulting structure for the object "tooth."

2.1.2 Deformable Models

A second project in range analysis involves the use of deformable surfaces to generate a 3-D approximation of range data. This work, performed by C. Liao and G. Medioni, builds on our earlier work in "B-Snakes." The user provides an initial simple surface, such as a cube, which is subject to internal forces (describing implicit

continuity properties such as tension and bending) and external forces which attract it toward the data points. The problem is cast in terms of energy minimization. We solve this non-convex optimization problem by using the well known Powell algorithm which guarantees convergence to a (possibly local) extremum and does not require gradient information. The variables are the positions of the control points. The number of control points is adaptively controlled. This methodology leads to a reasonable complexity and good numerical stability. We also provide a novel solution to the problem of subdividing a patch when the fit is bad. We show results on real range images to illustrate the applicability of our approach. The advantages of this approach are that it provides a compact representation of the approximated data, and lends itself to applications such as non-rigid motion tracking and object recognition. Currently, our algorithm gives only a C^0 continuous analytical description of the data, but due to the flexibility of our adaptive approach it should be upgraded to C^1 or C^2 easily. Figure 2 shows an example of the extraction of the surface description from the range data.

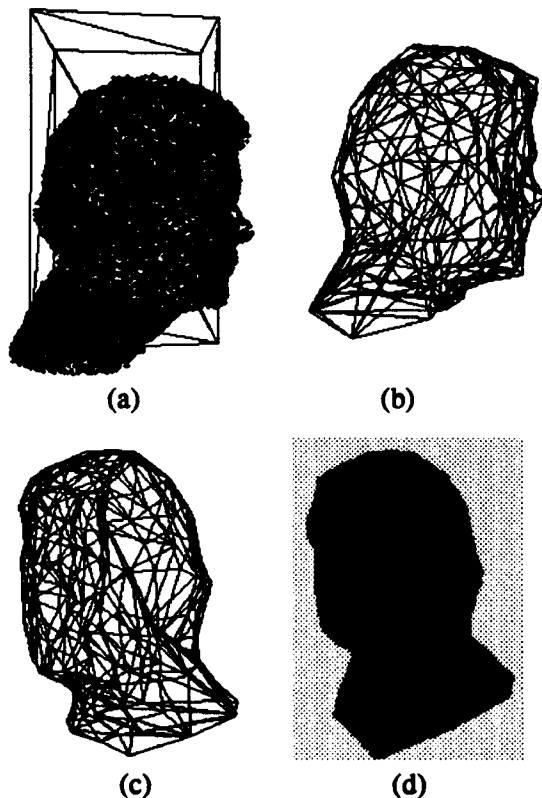


Figure 2 (a) shows one view of a head with 45524 sampled points, and the initial cube with 8 control points and 12 triangles. (b) and (c) are the fitting surface with 259 control points and 514 triangles. (d) shows the shaded surfaces after the third subdivision

2.1.3 Segmented Volumetric 3-D Descriptions

We address the problem of recovering segmented hierarchical volumetric descriptions of three dimensional shapes. In an earlier work [Rom & Medioni 1992][Rom & Medioni 1993], we have suggested a method (using SLS) for obtaining hierarchical axial descriptions of planar shapes, together with a decomposition of the shapes into their parts. Unfortunately, it is not straightforward to extend these methods to handle three dimensional shapes. This is because in the three dimensional space the SAT and SLS axes are, in general, not curves, but surfaces, leading to unnatural descriptions [Nackman 1985].

In this current work, performed by H. Rom and G. Medioni, we restrict ourselves to three types of parts: Convex blobs (or Ovoids, borrowing the terminology from Koenderink [Koenderink 1990]), Straight Homogeneous Generalized Cylinders (SHGCs [Shafer 1983]), and Planar Right Constant GCs (PRCGCs [Ulupinar 1991], planar axis and constant cross section). These components exhaust many of the man-made objects encountered on a normal basis. We suggest the use of properties of the parabolic curves (zero crossings of the Gaussian curvature) for recovering the cross sections and axes of the different parts. We advocate the use of the parabolic curves over the often used occluding contours, which are unstable in range data. We will assume that the shapes are C^2 continuous (i.e. the curvature is defined everywhere). We do not want to assume, as several authors do, that the parts are cut along a cross section or that a cross section is visible. Furthermore, we will not assume the existence of any discontinuity edges between parts. We believe that the case of parts joined discontinuously is the limiting case of the more general continuous case which we address.

Given the 3-D surface data, either from a CAD model, or from registered range images [Chen & Medioni 1992], or from a single range image, we first recover the parabolic curves on the surface. This requires the evaluation of the sign of the Gaussian curvature of the surface patches. It has been shown that this process is stable and reliable [Besl & Jain 1986] [Ponce & Brady 1987] [Fan *et al.* 1989]. The parabolic curves could be either on the surface of the individual parts, or on the border of the "glue" between parts. Note, that due to the transversality principle [Guillemin & Pollack 1974], there is almost always an anticlastic (negative Gaussian curvature) region between convex parts when they are joined. The parabolic curves on the parts we consider could be either meridians or cross sections of the SHGC and PRCGC parts (this has been shown for SHGCs [Ponce *et al.* 1989] and we have proven it for PRCGCs). Using simple tests we can hypothesize (or in many cases determine) the role of each parabolic curve. We can therefore segment the object into parts, and based on the

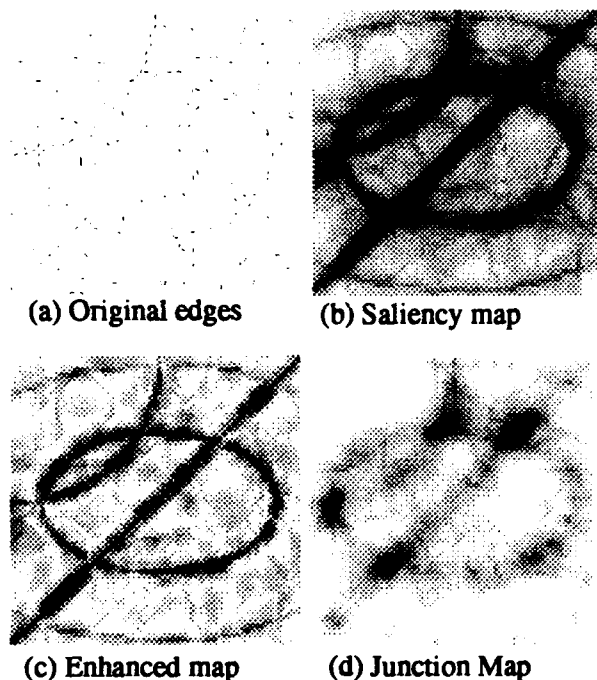


Figure 3 Steps in extracting the most salient features from an edge image.

properties of the specific parts, we can recover the axis of the parts from the meridians and cross sections.

One problem which remains is that some parts cannot be found until some other parts are removed. As in [Rom & Medioni 1992] and [Rom & Medioni 1993], we take an hierarchical strategy, in which, at each step, well defined parts are described and removed. Once these parts are removed, the next level parts can now be described. This process is efficient and produces a decomposition of the shape into its intuitive parts with a stable axial description of these parts.

2.2 Perceptual Grouping

We have started to develop a system for perceptual grouping based on the properties of collinearity and co-curvilinearity of partial contours [Guy & Medioni 1993]. The implementation is obtained by initiating an oriented vector field at each site detected by a low level detector. Structures which verify our constraints reinforce each other, and dominate other arrangements. Figure 3 shows a typical input, and the salient structures detected.

2.3 Shape Analysis from Monocular Images.

We have continued our effort in understanding how to infer shape from monocular images using contours. First we developed a theory of invariances of projected contours and how they can be used to infer 3-D shapes of a certain classes of surfaces [Ulupinar & Nevatia 1990, Ulupinar & Nevatia 1992, Ulupinar 1991]. In recent work, we have developed a system for generating volumetric 3-D shape descriptions from real images

containing Straight, Homogeneous Generalized Cylinders (SHGCs). The image may contain multiple, occluding objects and the objects may have surface markings. In working with real images, we must deal with problems of fragmented boundaries and many additional boundaries due to markings, shadows, highlights and noise. We use the expected properties of the desired contours to separate the two sets of properties and to complete the broken boundaries. Figure shows results on one example. Figure 4(a) shows the image, (b) the edges detected in it. Figure 4(c) shows the three reconstructed SHGCs from two viewpoints each. Further details of this process are given in another paper in these proceedings [Zerroug & Nevatia 1993a].

In continuation of this work, we are also studying the class of curved generalized cylinders with circular but changing cross-sections. In this case, we are unable to find invariants for the visible boundaries. However, we are able to find good *quasi-invariants* that show that commonly used ribbon descriptions are in fact, stable for such objects. Our future work will focus on compound objects that combine a number of primitives that we have analyzed in the past.

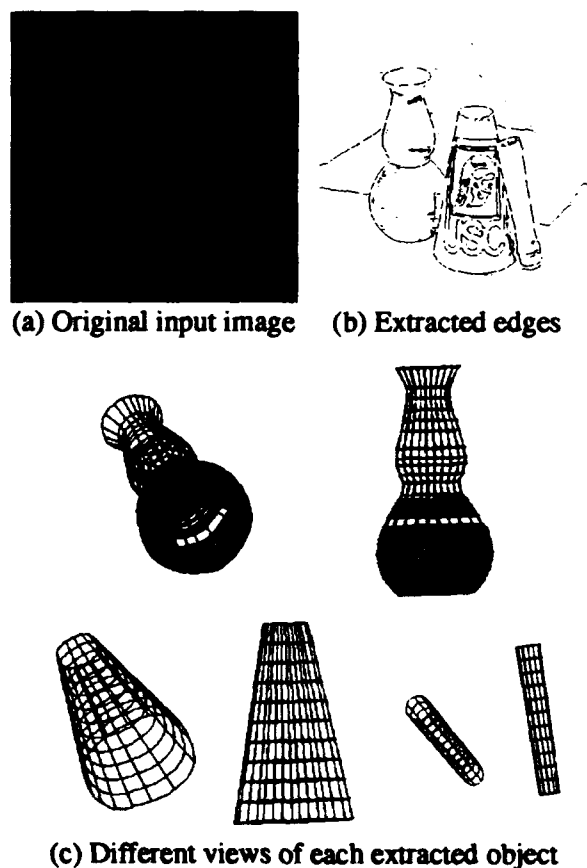


Figure 4 Generation of 3-D SHGC descriptions from edges extracted from an image.

3 MOTION ANALYSIS

We have advocated the use of multi-frame analysis for several years. In previous work, we developed a motion and structure estimation system based on so called *chronogeneous motion*, which includes uniform acceleration and constant angular velocity rotation and translation as special cases [Franzen 1991, Franzen 1992].

3.1 Integrated Motion System

In recent work using the motion estimation system, we have been building an integrated system that includes hierarchical feature extraction and matching and feedback of the Franzen 3-D motion estimation results to the feature matching process [Kim & Price 1993]. This enables the system to tolerate errors and differences in the feature extraction and matching processes by removing these inconsistent feature points from the later analysis. Figure 5 shows the first and last frames of an indoor image sequence (acquired from the University of Massachusetts) along with the reconstructed image-plane trajectories based on the 3-D analysis and the reconstructed top-down view of the scene.

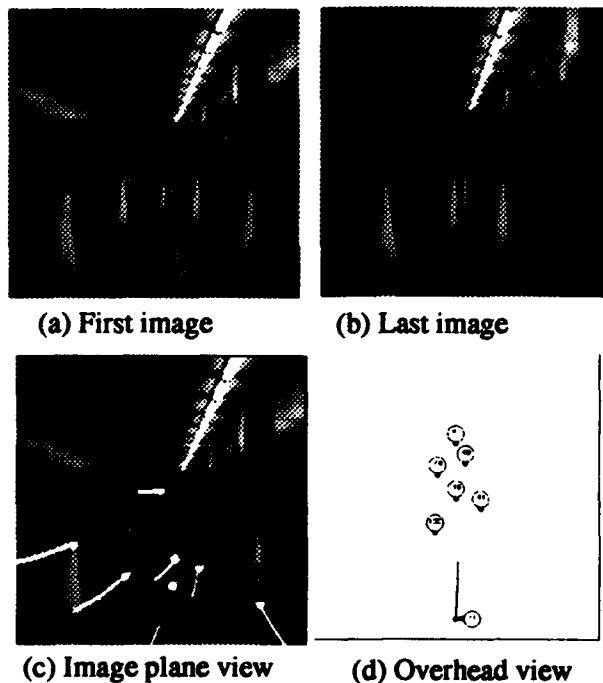


Figure 5 Images and reconstructed results for the cone sequence.

3.2 Mobile Platform

We have continued with our robot project using trinocular imagery for guidance. We are investigating robot navigation for situations where only *generic* maps are available, with one of the tasks being the generation of more complete maps. The visual navigation uses three views to improve the performance of the stereo system, both in speed and accuracy of the matching. Rather than producing a complete depth map we are concerned only with producing a "squeezed 3-D map" that shows corri-

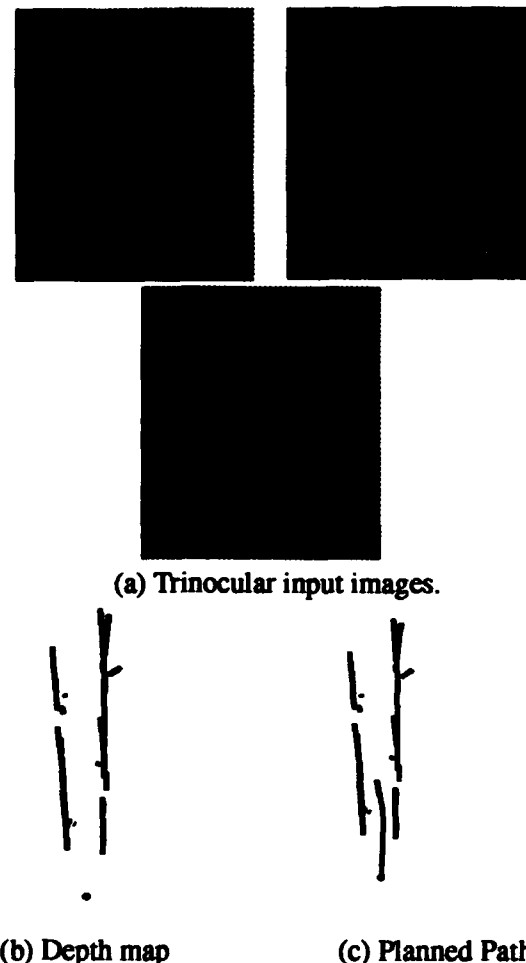


Figure 6 Trinocular vision results for mobile robot.

dor walls and obstacles in the hallway. Figure 6 shows the input three images for a hallway scene along with the resulting 3-D map and the planned path.

4 AERIAL IMAGE ANALYSIS

Most of our effort in this area is in support of the RADIUS program and is largely funded under a different contract though much of the basic technology has derived from our basic IU research effort. We have focused on the problem of 3-D object detection and description, particularly the buildings. Various RADIUS experiments with image analysts clearly illustrate the central role of buildings in a site.

4.1 Stereo Calibration

In work by M. Bejanin and G. Medioni, we have considered the problem of finding the geometric transformation between two perspective views of the same scene and the determination of epipolar lines, given a set of matched control points. Many methods exist and divide in two groups: linear methods and non-linear methods [Horn 1991]. Linear methods work with matrix operations, using the essential matrix that was introduced by Longuet-Higgins in 1981 [Hartley 1992] [Longuet-Hig-

gins 1981]. We have evaluated the performance of both types of methods using aerial images. We have also produced extensive comparisons between the two types of methods by testing them on synthetic random data. The main result of this comparative study is that the non-linear method seems to be more robust and reliable in the presence of noise. Also, the linear method has many degenerate cases where the transformation cannot be computed; among these is the "perfect" stereo case where both optical axis are parallel and perpendicular to the baseline. Finally, the result obtained on aerial scenes is not very accurate, as these scenes are nearly planar.

After the transformation between the two images has been computed from any of these methods, we show that it is possible to obtain, from the initial views, two images that are in parallel epipolar geometry. This is done by applying the rotation to the first image plane. This is equivalent to reprojecting the image, by keeping the same position in space for the principal point, but simply changing the direction of the principal ray. After having transformed the first image, our two images will lie on parallel image planes, but the scale will not necessarily be the same (that is to say the baseline is not necessarily parallel to both image planes). Therefore we still need to scale the first image to the size of the second image. We first transform the image coordinate system in both images such that the new x -axis and the new x' -axis are parallel to the baseline, we are then able to use the control points to scale the first image: since we want to get collinear epipolar images, two conjugate points should lie on the same line (in this new coordinate system), and so have identical y -coordinates. The results for real aerial images of the Ft. Hood site are shown in Figure 7 (flight parameters were used for computing the results).

4.2 Stereo for Buildings

The problem of building detection and description is difficult for a number of reasons. Aerial images tend to be highly complex with even simple buildings having many architectural details, surrounding trees and vehicles, nearby and aligned roads etc. These cause the low-level segmentation to produce highly fragmented results. Also, 3-D information is not explicit in 2-D images, but must be inferred somehow. We have developed two systems, one using stereo analysis, the other using shadow analysis to overcome both problems (of fragmentation and 3-D description).

Availability of stereo makes the task of inferring 3-D easier. However, we must still solve the problem of correspondence, a difficult one in this context, and infer surfaces from the partial stereo matches. In our system, we use scene features for correspondence. Our current system uses junction and line matches, though ability to match higher level features can also be incorporated. Surfaces are inferred from junction and line matches by

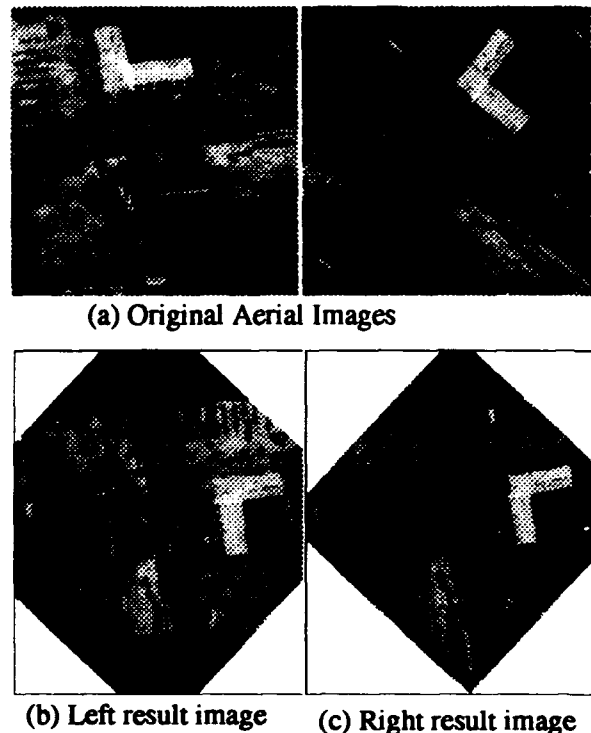


Figure 7 Two original images of the Ft. Hood area (top), and registered left and right transformed images in collinear epipolar geometry (bottom).

forming co-planar clusters and tracing structures among them. Our system can work with both overhead and "oblique" views. An example is shown in Figure 8. Note that the system produces a flat roof (as opposed to wavy surfaces typically found by stereo systems) and is able to detect the hole in the roof correctly. More examples and a detailed description of this system may be found in [Chung & Nevatia 1992b] and [Chung 1992].

4.3 Use of Groupings and Shadows

Another system we have developed uses only a single image, but uses shadows to verify presence of a building and estimate its height. This system is currently designed for overhead views; we are in the process of extending it for oblique views. Basically, the approach is to use perceptual grouping to hypothesize likely building-like structures. We assume that the buildings are rectangular or composition thereof. Thus, the hypothesis generation phase consists of forming rectangular hypotheses from fragmented line segments. We select among these hypotheses based on some geometrical analysis of overlap, containment and strength of the evidence forming the hypotheses. The selected hypotheses are then verified by examining whether they cast shadows in the appropriate ways. Figure 9 shows an example. Note that this building contains a number of parallel structures on top of the roof, making the hypotheses formation and selection process particularly difficult.

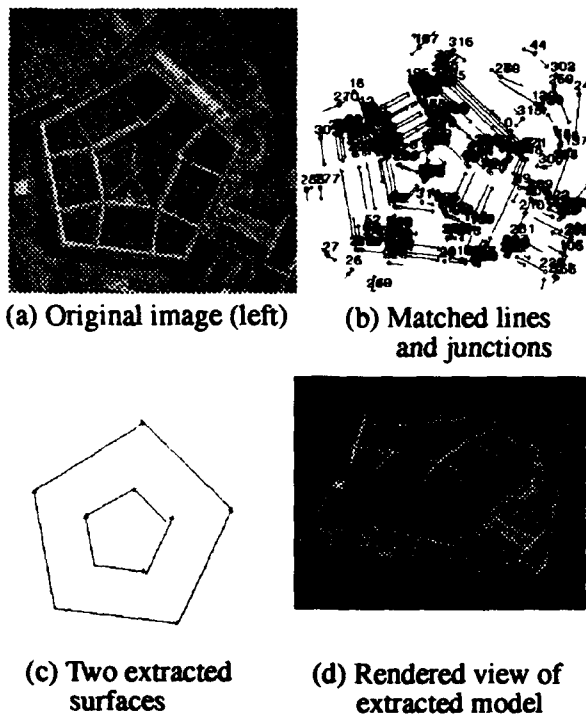


Figure 8 Analysis of buildings using stereo. The hierarchical matching uses lines and junctions to extract a set of surfaces, which are then shown as a rendered model.

Nonetheless, our system produces good results. More examples and a more complete description is given in another paper in these proceedings [Huertas *et al.* 1993].

5 KNOWLEDGE-BASED SYSTEMS

We have begun a new project in using standard knowledge representation technology for Image Understanding. Loom, a knowledge representation system developed at USC-ISI, provides a high level programming interface for Lisp and an environment for knowledge based system construction [MacGregor & Burstein 1991]. Since it is built on Lisp, it can be easily incorporated into our Lisp based environment and should be compatible with the Image Understanding Environment developments (which did not address the knowledge base aspects of image analysis). In past years we developed a system that uses domain knowledge to simplify the task of describing a scene [Huertas *et al.* 1990]. This original system encoded the knowledge about airports in the programs.

We chose to apply Loom to the airport analysis problem so that we could address the knowledge representation issues separate from the image understanding issues. This is possible since the basic analysis programs already exist and the incorporation of Loom into the analysis can proceed in an incremental fashion, first with the use of Loom as a representation system for the

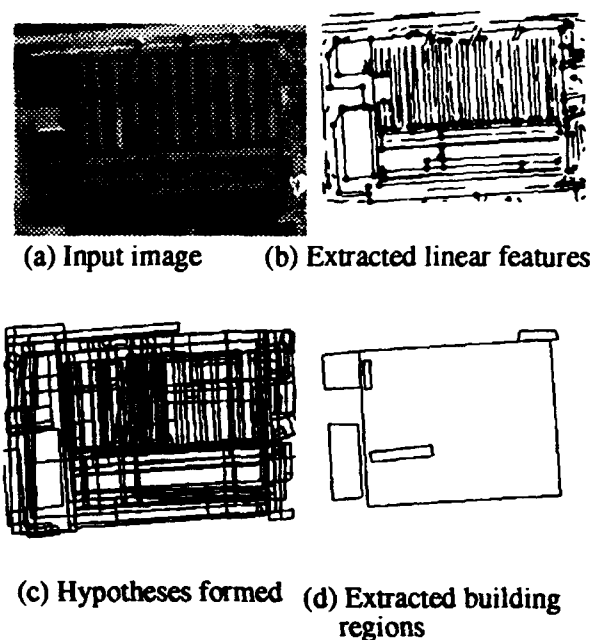


Figure 9 Building detection using shadows for verification. Using the lines and junctions, the buildings are extracted and verified using shadows.

generic descriptions of airports and for particular instance that we are analyzing. Then we will begin to use the other descriptive and deductive capacities for finding and analyzing the runway markings that are used for the final verification and positioning of the runways. Loom provides a means to describe the other objects that may be present in the scene (taxiways, building, aircraft, etc.) and their relations to the other objects.

Our first goal is to evaluate the capabilities of Loom for high level Image Understanding and then to use Loom for generic high level descriptions of complex objects, and to use these descriptions for building other analysis systems.

6 PARALLEL PROCESSING

We have studied parallel implementations of several high-level algorithms, such as relaxation labelling and graph matching. Our recent work has looked at the problem of geometric hashing, which is used for a variety of matching problems [Stein & Medioni 1992]. In earlier parallel implementations the number of processors was independent of the size of the scene but depended on the size of the model database. In this work we have designed new parallel algorithms for both the MasPar and Connection Machine architectures which improve on the number of processors and improve the overall performance. Details of this work are given in [Khokhar & Prasanna 1993].

7 ACKNOWLEDGMENTS

This paper represents the work of many different current and former students, visitors, staff members and associated faculty. Y. Chen is working on the integration from multiple views. C. Liao is working on deformable models. H. Rom is working on segmented volumetric 3-D descriptions. G. Guy is working on perceptual grouping. M. Zerroug is working on shape analysis from monocular contours. R. Chung completed a thesis on hierarchical features in stereo including detecting buildings in stereo images. F. Stein completed a thesis on matching using structural indexing. Y. Kim is working on the integrated motion system. D. Kim is working on the mobile platform and trinocular vision. M. Bejanin is working on the stereo calibration problem. A. Huertas and C. Lin are working on the aerial image analysis. A. Khokhar and V. Prasanna are working on the analysis of parallel algorithms.

REFERENCES

- [Besl & Jain 1986] P. J. Besl and R. C. Jain, "Invariant Surface Characteristics for 3D Object Recognition in Range Images," *Computer Vision Graphics and Image Processing*, Vol. 33, pages 33-80, 1986.
- [Chen & Medioni 1992] Y. Chen and G. Medioni, "Object Modeling by Registration of Multiple Range Images," *Image and Vision Computing*, Vol. 10, No. 3, pages 145-155, April 1992.
- [Chen & Medioni 1993] Y. Chen and G. Medioni, "Surface Level Integration of Multiple Range Images," in *Proc. DARPA Image Understanding Workshop*, Washington, DC, April 1993.
- [Chung & Nevatia 1992] C.-K. R. Chung and R. Nevatia. Recovering LSHGCs and SHGCs from stereo. In *Proceedings of the DARPA Image Understanding Workshop*, pages 401-407, San Diego, CA, January 1992.
- [Chung & Nevatia 1992b] C.-K. R. Chung and R. Nevatia, "Recovering Building Structures from Stereo," In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, pages 64-73, Palm Springs, CA, December 1992.
- [Chung 1992] C.-K. R. Chung, *Deriving 3-D Shape Descriptions from Stereo Using Hierarchical Features*, Ph.D. Thesis, University of Southern California, November 1992.
- [Fan et al. 1989] T. J. Fan, G. Medioni, and R. Nevatia, "Recognizing 3-D Objects Using Surface Descriptions," *IEEE Trans. on PAMI*, Vol. 11, No. 11, Nov. 1989, pages 1140-1157.
- [Franzen 1991] W. Franzen, "Structure and Motion from Uniform 3D Acceleration," In *Proceedings of the Workshop on Visual Motion*, pages 14-20. IEEE Computer Society, Oct 7-9, 1991.
- [Franzen 1991] W. Franzen, *Structure from Chronogeneous Motion*, Ph.D. thesis, University of Southern California, May 1991, IRIS Technical Report 266.
- [Franzen 1992] W. Franzen "Structure from Chronogeneous Motion: A Summary," In *Proceedings of the DARPA Image Understanding Workshop*, San Diego, CA, January 1992.
- [Guillemin & Pollack 1974] V. Guillemin and A. Pollack, *Differential Topology*, Prentice Hall, 1974.
- [Guy & Medioni 1992] G. Guy and G. Medioni, "Perceptual Grouping Using Global Saliency Enhancing Operators," In *Proceedings of the International Conference on Pattern Recognition*, pages 99-104, The Hague, Holland, August 1992.
- [Guy & Medioni 1993] G. Guy and G. Medioni, "Inferring Global Perceptual Contours from Local Features," in *Proceedings of the DARPA Image Understanding Workshop*, Washington, DC, April 1993.
- [Hartley 1992] R. I. Hartley, "Calibration of Cameras Using the Essential Matrix," in *Proceedings of the DARPA Image Understanding Workshop*, pages 911-915, San Diego, CA, January 1992.
- [Horn 1991] B. K. P. Horn, "Relative orientation revisited," in *Journal of the Optical Society of America A.*, Vol. 8, NO 10, pages 1630-1638, October 1991.
- [Huertas et al. 1990] A. Huertas, W. Cole, and R. Nevatia, "Detecting Runways in Complex Airport Scenes," *Computer Vision, Graphics, and Image Processing*, Vol. 51, No. 2, pages 107-145, August 1990.
- [Huertas et al. 1993] A. Huertas, C. Lin and R. Nevatia, "Detection of Buildings from Monocular Views of Aerial Scenes Using Perceptual Grouping and Shadows," in *Proceedings of the DARPA Image Understanding Workshop*, Washington, DC, April 1993.
- [Kim & Nevatia 1993] D. Kim and R. Nevatia, "Indoor Navigation without a Specific Map," in *Proceedings Workshop on Intelligent Systems*, Pittsburgh, PA, 1993.
- [Kim & Price 1993] Y. C. Kim and K. Price, "Improved Correspondence in Multiple Frame Motion Analysis," In *Proceedings of the DARPA Image Understanding Workshop*, Washington, DC, April 1993.
- [Khokhar & Prasanna 1993] A. Khokhar and V. Prasanna, "Scalable Data Parallel Geometric hashing: Experiments on MasPar MP-1 and on Connection Machine CM-5," In *Proceedings of the DARPA Image Understanding Workshop*, Washington, DC, April 1993.
- [Koenderink 1990] J.J. Koenderink, *Solid Shape*, M.I.T. Press, Cambridge, MA, 1990.

- [Liao & Medioni 1993] C. Liao and G. Medioni, "Adaptive Surface Approximation of a Cloud of 3D Points," USC report, submitted for publication.
- [Longuet-Higgins 1981] H.C. Longuet Higgins, "A computer algorithm for reconstructing a scene from two projections," in *Nature*, Vol. 293, pages 133-135, September 1981.
- [MacGregor & Burstein 1991] R. MacGregor and M. Burstein, "Using a Description Classifier to Enhance Knowledge Representation," *IEEE Expert*, Vol 6, No. 3, pages 41-46, June 1991
- [Nackman 1985] L. Nackman and S. Pizer, "Three-Dimensional Shape Description Using the Symmetric Axis Transform," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 7, No. 2, March 1985, pages 187-202.
- [Parvin & Medioni 1992] B. Parvin and G. Medioni, "B-rep from Unregistered Multiple Range Images," In *Proceedings of the IEEE Conference on Robotics and Automation*, Nice, France, May 1992.
- [Ponce & Brady 1987] J. Ponce and M. Brady, "Towards a Surface Primal Sketch," in *Three Dimensional Machine Vision*, T. Kanade (ed.), Kluwer Academic, New York, pages 195-239, 1987.
- [Ponce *et al.* 1989] J. Ponce, D. Chelberg and W.B. Mann, "Invariant Properties of Straight Homogeneous Generalized Cylinders and Their Contours," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 11, No. 9, pages 951-966, 1989.
- [Reinhart 1991] C. Reinhart, *Specifying Parallel Processor Architectures for High-Level Computer Vision Algorithms*, Ph.D Thesis, University of Southern California, December 1991.
- [Rom & Medioni 1992] H. Rom and G. Medioni, "Hierarchical Decomposition and Axial Shape Description," In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 49-55, Champaign, IL, June 1992.
- [Rom & Medioni 1993] H. Rom and G. Medioni, "Hierarchical Decomposition and Axial Shape Description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, to appear 1993.
- [Shafer 1983] S.A. Shafer, "Shadow Geometry and Occluding Contours of Generalized Cylinders," Technical Report CS-83-131, Carnegie Mellon University, May 1983.
- [Stein & Medioni 1992] F. Stein and G. Medioni, "Structural Indexing: Efficient Three Dimensional Object Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 14, No. 2, pages 125-146, February 1992.
- [Ulupinar & Nevatia 1990] F. Ulupinar and R. Nevatia, "Recovering Shape from Contour for SHGCs and CGCs," In *Proceedings of the DARPA Image Understanding Workshop*, pages 544-556, Pittsburgh, PA, September 1990.
- [Ulupinar & Nevatia 1992] F. Ulupinar and R. Nevatia, "Recovery of 3-D Objects with Multiple Curved Surfaces from 2-D Contours," In *Proceedings of the DARPA Image Understanding Workshop*, pages 627-633, San Diego, CA, January 1992.
- [Ulupinar 1991] F. Ulupinar, *Perception of 3-D Shape from 2-D Image of Contours*, Ph.D. Thesis, University of Southern California, August 1991.
- [Zerroug & Nevatia 1993] M. Zerroug and R. Nevatia, "Scene Segmentation and Volumetric Descriptions of SHGCs from a Single Intensity Image," In *Proceedings of the DARPA Image Understanding Workshop*, Washington, DC, April 1993.
- [Zerroug & Nevatia 1993b] M. Zerroug and R. Nevatia, "Quasi-Invariant Properties and 3-D Shape Recovery of Non-Straight, Non-Constant Generalized Cylinders," In *Proceedings of the DARPA Image Understanding Workshop*, Washington, DC, April 1993.

Image Understanding at the University of Rochester

Randal C. Nelson
Christopher M. Brown
Computer Science Department
University of Rochester
Rochester, NY 14627

Abstract

Machine Vision Research at the University of Rochester centers around the concept of active or behavioral vision. Vision is considered as an interactive process between a system and its environment, and this relationship is explicitly invoked to provide constraints to make machine-vision applications tractable. Work in this area has focussed on four main areas: integration of visual and motor control, learning for robot skill and visual model acquisition, task dependent allocation of computational and physical resources, and parallel real-time systems integration and support. We also continue to develop real-time low-level primitives for integration into larger systems.

1. The Laboratory

The Computer Vision Laboratory at the University of Rochester is set up to study active vision in a very real sense - by interacting with the physical world. Robot hardware includes a binocular head with three external degrees of freedom, an Utah/MIT hand, and two Puma robot arms for moving the hand and head. Computational hardware includes a large collection of Maxvideo boards for performing real-time image processing, an eight node Silicon Graphics Multiprocessor, an array of eight transputer T805 TRAMS, and various coordinating workstations. Several improvements have been made to the lab in the last year. The head has been modified to provide color images and internal degrees of freedom, including focus and zoom. The second Puma arm was acquired to give us the capability to move the head and hand cooperatively. The Puma arms were converted to run RCCL instead of VAL. This gives us more flexibility, and provides a much needed increase in speed. We have obtained an EXOS exoskeleton for measuring human hand movements during manipulation. This replaces the DataGlove as an input device for the Utah hand, and provides values that are far more stable and repeatable. We have also founded a Virtual Reality lab for investigating psychophysical aspects of vision. This lab contains eye and head trackers that will be incorporated into a helmet-mounted stereo video system. The equipment will be used to study human visual strategies during the performance of various tasks. Recent research

at the lab was described in a dedicated special issue of the International Journal of Computer Vision [Nelson 1991a and sequel]. The animate/behavioral vision approach is also described in [Ballard et al. 1992, Ballard 1991, Moraff and Brown 1992].

2. Integration of Visual and Motor Control.

Previous work on low-level gaze control culminated with a successful integration of vergence and tracking that permitted the robot head to track an object moving in a cluttered scene binocularly in real time. The system combined foveal regions of interest, a binocular disparity filter, and predictive tracking to obtain the demonstrated performance [Brown 1991a, Brown and Coombs 1991, Brown et al 1992, Coombs and Brown 1992, Grosso and Ballard 1992, Soong and Brown 1992].

A new initiative by Nelson addresses hand-eye coordination using local linear models [Nelson 1993b]. The general idea is to relate changes in object appearance to a set of one-parameter manipulations or "twiddles" by means of a generalized Jacobian. We represent the appearance of an object by a vector x of scalar quantities, which could be the image coordinates point features, segment, curve or blob parameters, or even colors. We also have a vector y of qualitative one-parameter manipulations or "twiddles", which could involve both rigid motions and nonrigid deformations. For a particular pose of the object, we can describe the change in appearance of the object under a small manipulation by a *motor-visual Jacobian* formed from the partial derivatives of the feature values with respect to the various manipulations. A nice property of this Jacobian is that it can be determined interactively, i.e., learned. The robot just picks up an object, looks at it, makes small manipulations along its basis axes, and observes the changes in appearance.

The motor-visual Jacobian can be used to implement a tight version of a principle views 3-D recognizer by storing with each view the motor-visual Jacobian corresponding to that pose and some set of basis manipulations. Given a match hypothesis from some flexible matching process, a difference vector can be computed giving the discrepancy between stored

representation and the measured values of the corresponding features. We then use a well conditioned pseudoinverse computed from the singular value decomposition of the Jacobian, to find the manipulation that comes closest (e.g. in a least squares sense) to producing the observed view. By checking the difference between this solution and the actual appearance, we can tell whether what we see is consistent with what is encoded about how the appearance of the object can change. This procedure allows an accurate match to be made without an explicit 3-D model, and using only visually obtained information.

The same mechanism used for recognition can be reversed to visually control manipulations. The basic idea is to experimentally determine the visual effect of qualitative manipulations of an object and represent them, as previously, by a motor-visual Jacobian. If we know the appearance of the object in the goal condition, we can compute a difference vector and solve for the best manipulation vector using the pseudoinverse as before. Executing this manipulation, we will effect the goal configuration within the accuracy of the linear approximation. The primary advantage of the proposed technique is that it can be used with a manipulator and visual system for which a good physical model does not exist.

3. Learning Applications

Learning has become increasingly important in our research, where we define learning as the process of obtaining some of the information needed to specify a complex machine system automatically, either via interaction with the world, or using a search process to find internal parameters that produce some more loosely specified behavior. Along the theoretical axis, Whitehead finished his work on using variations of reinforcement learning to acquire execution models for joint foveal and manipulator control for goal-state specified tasks in complex environments [Ballard et al. 1992, Ballard and Whitehead 1992]. Ballard and deSa continued work on self-teaching variants of competitive learning [DeSa and Ballard 1992]. The basic idea is to use the non-flat nature of joint probability conditional density functions between different sensory modalities to drive the segregation of classes when the information in a single modality is not appropriately biased to drive a clustering process. Brown also pursued topics in geometric invariance, which allow recognition models to be automatically acquired [Brown 1991b, Brown et al. 1992]. On the applications side, two projects are ongoing. One, by Pook and Ballard attempts to derive high-level execution models for complex manipulation tasks by observing teleoperated sequences. The other, by Schneider and Brown, attempts to learn motor control strategies for parameterizable actions that optimize some cost function. These are described briefly below.

3.1 Tele-assisted Manipulation

Teleoperation, wherein a robot mimics the actions of a human, offers the means to learn action sequences. If one records the robot state under teleoperated control then one can later replay the sequence of moves to duplicate the performance. The problem with such replays is that they are open-loop; the controller cannot respond to even minor changes in the environment. Pook and Ballard are developing the idea of tele-assistance, a form of control that combines the learning abilities of teleoperation with closed-loop autonomous control, to address the common latency and accuracy problems of teleoperation [Pook and Ballard 1992a, 1992b, Chu 1992]. For example, as an operator slides a tool across a surface, a closed-loop controller can maintain steady robot velocity and contact with the surface even if the operator action is jerky.

The first step in tele-assistance is to identify a primitive actions as they are performed. Using teleoperation of the Utah-MIT hand to perform tasks such as flipping eggs with a spatula, Pook has successfully identified action primitives such as grasping, carrying, pressing, and sliding in contact, from characteristic temporal patterns of joint forces. This was done using learning vector quantization to generate codebook patterns from a training set of actions. The codebook patterns were then used in conjunction with a hidden Markov model for transitions between actions to identify the action primitives in the context of a larger task.

The second step in teleassistance is to produce the closed loop controls that implement the action primitives. Our current approach is based on the use of parameterized behaviors that reduce the redundant degrees of freedom in the hand to a manageable level via a task-dependent linking mechanism. We have had some success already in producing such parameterized behaviors. Pook and Ballard will continue to pursue this step over the upcoming year.

3.2 Robot Skill Learning

Schneider and Brown are working on the topic of robot skill learning [Schneider and Brown 1992]. Skills are parameterized open-loop behaviors useful for tasks for which the delay of closed-loop control can not be tolerated, or for which no intermediate performance information is available. Throwing a ball at a target is an example of the latter. A skill is termed "n-dimensional" when the task is parameterized by n desired output values.

Robot skill learning can be modeled as a function approximation problem. The learning algorithm needs to find a mapping from n desired output values to the space of possible robot control trajectories. These input control trajectories may be sequences of joint positions, velocities, or torques. Typically the mapping from robot control signal to task output is redundant and the inverse to be learned should be optimized according to a cost metric. Function approximation techniques can

be divided into global and local methods.

In global approximation techniques, a variable functional form is specified in terms of parameters that affect the entire domain of the mapping. The learning algorithm must find the parameter settings that optimize a cost function. Linear models are easy to deal with, but restrictive when the task is non-linear, as is the case in most robot control problems. We therefore adopted a modified linear model in which learning was done in a linear space, but a set of non-linear basis functions was used to convert points in the output space into robot trajectories. Experiments with a one-dimensional throwing task (the single task parameter is distance to throw) were done in simulation. The results showed that global function approximation can be used for one-dimensional throwing. Performance is highly dependent on choice of basis functions and robot control schemes (joint velocities, torques, or spring-based control). The largest drawback is the number of robot executions necessary to fit the model, which increased dramatically with the number of task dimensions. This led us to consider local function approximation methods.

Canonical local function approximation methods include neural networks, nearest neighbor classifiers, table lookup, and radial basis functions. Schneider used a form of interpolated table lookup modified to handle a high (18) dimensional input space efficiently to learn a two-dimensional (x and y landing position of the ball) throwing task in simulation. Experiments were run with 1-d and 2-d throwing to compare the brute force search of standard table lookup and the new algorithm. The results showed a 55% to 80% reduction in the cost metric (includes accuracy and control effort) for the same number of robot executions. The experiments also showed that the local method could outperform the global method even when it had the benefit of good basis functions. An additional feature of the new algorithm is its ability to automatically extend its range of performance as it acquires the skill. Neither standard table lookup nor the global method allows this.

4. Intelligent Resource Allocation

One of the major driving forces behind research into active vision is the need for some sort of sophisticated control over the allocation of resources, both physical and computational [Ballard and Brown 1992, Brown 1992b]. As an example of the first case, eyes, hands, and other sensors and manipulators generally can't be everywhere at once, and moreover, may require considerable time or energy to transfer between states. As an example of the second, it is generally not necessary for a vision system to identify everything identifiable in a scene, and is probably a waste of resources to do so in most applications. This brings up the issue of visual attention, where resources are allocated on the basis of what is most likely to benefit the task at hand. Two current projects address the issue of visual attention in various ways.

4.1 Resource Allocation Using Bayes Nets

Rimey and Brown have addressed the problem of intelligent resource allocation using a Bayes net formalism [Rimey and Brown 1991, 1992a, 1992b, 1993]. The general problem is to control a computer vision system that has a repertoire of actions so that it achieves some goal in minimal time. In particular we want to accomplish visual tasks efficiently by using knowledge about the scene domain and about available visual and non-visual operators. Efficiency comes from processing the scene only where necessary, to the level of detail necessary, and with only the necessary operators.

TEA-1 is a general purpose selective computer vision system that attempts to accomplish these goals using Bayes nets for representation and a maximum expected utility rule to make decisions about what action to take. It is both a prototype system and an open-ended research tool for studying control of selective perception. The basic constraint assumed for the vision system is a pointable, multiresolution sensor that cannot view the whole scene at once. The problem is how best to utilize this sensor to achieve a particular goal. We have performed extensive experiments both in simulation and in the lab.

A number of modifications have recently been made to improve the performance of the system. First, actions have been split into two types: visual actions that process image data and camera movement actions. Previously all camera movements were integrated into each visual action. This both reduces the number of cases that must be considered, and decouples the analysis. Second, all the decision algorithms have been extended so they can be based on an expected value sample information (EVSI) measure. The main advantage of using an EVSI measure is that values and costs throughout the system are formally consistent, and in practice there are fewer coefficients that need to be adjusted. Third, A new algorithm that is based on a brute-force state-space search was added for deciding which visual and camera movement actions to execute. In addition a formal definition of the T-world domain and a software system for creating and solving T-world problems, has been developed. This enables analysis of a variety of factors that affect the performance of a selective perception system.

4.2 Searching Cluttered Areas

Object search is one area in which animate vision strategies can have a high payoff [Swain et al. 1992]. When searching for an object, it can be advantageous to make use of the spatial relationships in which it commonly participates. Searches that do this, which we call indirect searches, can be modeled as two-stage processes that first find an intermediate object that commonly participates in a spatial relationship with the target object, and then look for the target in the restricted region specified by this relationship. Using this model, Wixson has determined that over a wide range of situations, indirect searches improve efficiency by

factors of 2 to 8 [Wixson and Ballard 1991, Wixson 1992] To exploit such spatial relationships, it is necessary to have mechanisms for selecting camera positions. Since large objects that are suitable intermediate objects are usually cluttered, we need general-purpose, yet computationally efficient mechanisms for detecting these occluded areas and bringing them into view.

Occluding edges indicate areas that cannot be viewed from the current viewpoint, and Wixson believes that sparse information about occluding edges can be used to construct simple but efficient search mechanisms. Recently, he has developed an algorithm for detecting occluding edges in stereo or motion pairs. The algorithm works by searching for matches, in the right image, for the regions to the left and right of a left-image edgel. The algorithm is based on that of Toh and Forrest but extends that work in several ways. First, it adds an algorithm for automatically selecting an appropriate size for the correlation windows used to detect the occlusions. Second, it identifies a situation in which simply examining the match values is insufficient to determine whether an edgel is a surface marking or an occlusion. Finally, it adds a post-processing step that eliminates some falsely detected occlusions. Work on procedures for searching cluttered areas using this algorithm is currently underway.

5. Parallel Systems Support

Systems support for parallel processing applications in AI is an ongoing theme at Rochester [Bianchini and Brown 1992, Marsh et al. 1992, Weems et al. 1991] Currently, Robert Wisniewski is working on systems support for real-time parallel intelligent applications. There is a growing interest in designing such applications. Currently there does not exist a good software platform upon which to build these applications. The goal is to design a general system that allows for the necessary intelligent 'hooks' or information exchange between the high level application and the underlying system. Previous work has developed systems with good information exchange at the single application level, but general parallel environments with good interfaces between high and low levels do not yet exist.

We are using a parallel shepherding application developed on an eight node SGI multiprocessor to develop and test our proposed run-time environment. We believe this application is representative of AI applications needing to function in a real world environment. The goal is to keep as many individually moving objects confined on a table top as possible. Different planners using different amounts of time devise strategies for a robot arm manipulator using the visual input of an overhead camera.

Our work has examined the effectiveness of our run-time environment in choosing the appropriate planner for a particular dynamic internal state of the system. The run-time environment is aware of the application's goals by a set of shared data structures. This permits the execution level to maintain environmental

independence while still allowing information to be propagated throughout the system. The run-time environment also keeps track of the state of the underlying processors and system state. When the application indicates it wants to solve a new goal, the run-time environment uses the information it maintained about the system (e.g. about deadlines and accuracy requirements) and combined it with knowledge of the structure of the application to choose an appropriate planner. Our results indicate that this strategy works better than having a single monolithic planner.

6. Motion Recognition

Polana and Nelson have worked on robustly computable motion features that can be used directly as a means of recognition. The underlying motivation is the observation that, for objects that typically move, it is frequently easier to identify them when they are moving than when they are stationary [Nelson 1991]. Specifically, the goal is to design, implement and test a general framework for recognizing both distributed motion activity on the basis of temporal texture and complexly moving compact objects on the basis of their action. This recognition approach contrasts with the reconstructive approach that has typified most prior work on motion. The proposed work has practical applications in monitoring and surveillance, and as a component of a sophisticated visual system.

For the first phase of the project image sequences containing temporal textures were analyzed. Real imagery was used as the prime source of test data. The normal flow field of the motion between successive frames was used as the basis for recognizing the temporal textures. Several features were extracted from the normal flow field and techniques analogous to the statistical methods of gray-level texture classification were applied to successfully classify scene regions containing non-rigid motion [Nelson and Polana 1992].

For the second phase, image sequences containing a single periodic activity were analyzed in order to tag and track objects exhibiting periodic movement. Identifying such motion is important since it indicates a situation where a structural classification technique would be more appropriate than a temporal texture method. A Fourier Transform based technique was developed that successfully distinguished periodic activities such as walking, exercising, rotating machinery etc. from non-periodic motion, and tracked the region of periodic activity against cluttered backgrounds. Both stationary activity, and periodic activity resulting in translation of the actor can be identified [Polana and Nelson 1993]. The technique can be generalized to arbitrarily moving objects exhibiting periodic activity.

The current task is recognition of periodic activities after successful detection using the periodicity detection described above. A combination of normal flow field features and Fourier domain features will be used to characterize the nature of periodicity exhibited by

different activities. These will be applied specifically to recognize human walking, animal gaits, and periodically operating machinery. The techniques will also be examined for their invariance with respect to spatial scaling and viewing angle.

7. Line Segment Extraction

Finding lineal features in an image is an important step in many object recognition and scene analysis procedures. Nelson has developed a new method of extracting lineal features from an image using extended local information to provide robustness and sensitivity [Nelson 1993a]. The method utilizes both gradient magnitude and direction information, and incorporates explicit lineal and end-stop terms. These terms are combined non-linearly to produce an energy landscape in which local minima correspond to lineal features called *sticks* that can be represented as line segments. A gradient descent (stick-growing) process is used to find these minima.

More specifically, suppose there exists a matching criterion by which any line segment can be compared against an image and given a score. Then conceptually, a good set of line segments could be found by finding, from all possible segments, the one that produces the best score, nullifying the effect of the image components contributing to it, and repeating, until enough segments had been found. The main practical problems with this method are making it efficient, since it is clearly impractical to look through all possible segments multiple times, and designing an appropriate matching measure. The new matching measure utilizes a non-linear combination of separate convolutions with line-like and end-stop templates that provides for growth along a lineal feature, but stops the growth when strong evidence of a termination is encountered. This cannot be achieved with a single convolution measure.

When compared against two other methods of line segment detection, one based on edgel linking, and the other on support regions of similar gradient direction, the stick growing method exhibits improved gap crossing abilities, and is better able to extract long, poorly defined features, especially in cluttered images. The method gives sufficiently good results in images of objects having strong linear features to permit the intermediate level representation obtained to be used for recognition.

References

Ballard, D.H., "RISC models of visual behaviors," invited paper, *Proc., 3rd Int'l. Forum on the Frontier of Telecommunications Technology*, Tokyo, November 1991.

Ballard, D.H. and C.M. Brown, "Principles of animate vision," *Computer Vision, Graphics, and Image Processing 56-IU* (Special Issue on Active Vision), 1, 3-

21, July 1992; to appear, Y. Aloimonos (Ed.). *Active Vision*.

Ballard, D.H., C.M. Brown, and R.C. Nelson, "Image understanding research at Rochester," *DARPA Image Understanding Workshop*, 109-116, San Diego, CA, January 1992.

Ballard, D.H., M.M. Hayhoe, F. Li, and S.D. Whitehead, "Hand-eye coordination during sequential tasks," *Proc., Phil. Trans. Royal Society of London B*, London, March 1992.

Ballard, D.H. and S.D. Whitehead, "Learning visual behaviors," in H. Wechsler (Ed.). *Neural Networks for Machine Perception*, Vol. 2. Boston, MA: Academic Press, 1992.

Bianchini, R. and C.M. Brown "Parallel genetic algorithms on distributed-memory architectures," TR 436, Computer Science Dept., U. Rochester, August 1992.

Brown, C.M., "An empirical investigation of differential invariants," in J.L. Mundy and A.W. Zisserman (Eds.). *Computational Invariants for Vision*. Cambridge, MA: MIT Press, 215-227, 1992a.

Brown, C.M., "Gaze behaviors for robotics," invited paper, in A. Sood (Ed.), *Active Perception and Robot Vision* (Proc., NATO-ASI Symp. on Active Perception and Robot Vision, Maratea, Italy, July 1989). Springer-Verlag, August 1991a.

Brown, C.M. "Issues in selective perception," *Proc., 11th IAPR Int'l. Conf. on Pattern Recognition*, 21-30, The Hague, IEEE Computer Society Press, September 1992b.

Brown, C.M., "Numerical evaluation of differential and semi-differential invariants," TR 393, Computer Science Dept., U. Rochester, August 1991b.

Brown, C.M. and D.J. Coombs, "Notes on control with delay," TR 387, Computer Science Dept., U. Rochester, August 1991.

Brown, C.M., D.J. Coombs, and J. Soong, "Real-time smooth pursuit tracking," in A. Blake and A. Yuille (Eds.) *Active Vision*. Cambridge, MA: MIT Press, 123-136, 1992.

Brown, C.M., H. Durrant-Whyte, J. Leonard, B. Rao, and B. Steer, "Distributed data fusion using Kalman filtering: A robotics application," in M.A. Abidi and R.C. Gonzalez (Eds.). *Data Fusion in Robotics and Machine Intelligence*. Academic Press, 267-310, 1992.

Chu, Man-Wai, "Polhemus coordinates and Polhemus-Puma conversion," TR 427, Computer Science Dept., U. Rochester, June 1992.

Coombs, D.J. and C.M. Brown, "Real-time smooth pursuit tracking for a moving binocular head," *Proc., IEEE Conf. on Computer Vision and Pattern Recognition*, 23-38, Champaign, IL, June 1992.

de Sa, V.R. and D.H. Ballard, "Top-down teaching enables task-relevant classification with competitive learning," *Proc., Int'l. Joint Conf. on Neural Networks*, Baltimore, June 1992.

Grosso, E. and D.H. Ballard, "Head-centered orientation strategies in animate vision," TR 442, Computer Science Dept., U. Rochester, October 1992.

Marsh, B.D., C.M. Brown, T.J. LeBlanc, M.L. Scott, T.G. Becker, P.Ch. Das, J. Karlsson, and C.A. Quiroz, "Operating system support for animate vision," *Journal of Parallel and Distributed Computing* 15.2, 103-117, June 1992.

Moraff, H. and C.M. Brown, "Vision as process: A joint NSF/ESPRIT research project," *SPIE Robotics and Machine Perception Newsletter*, July 1992.

Nelson, R.C., "Introduction: Vision as intelligent behavior: An introduction to machine vision at the University of Rochester," *Int'l. J. of Computer Vision* 7, 1 (Special Issue), 5-9, 1991a.

Nelson, R.C., "Qualitative detection of motion by a moving observer," *Int'l. J. of Computer Vision* 7, 1 (Special Issue), 33-46, 1991b.

Nelson, R.C. and R. Polana, "Qualitative recognition of motion using temporal texture," *Computer Vision, Graphics, and Image Processing* 56-IU (Special Issue on Active Vision), 1, 78-89, July 1992.

Nelson, R.C., "Finding line segments by stick growing," submitted for journal publication 1993a.

Nelson, R.C., "Learning from Manipulation: Merging Seeing and Doing in Three Dimensions", TR 426 Computer Science Dept., U. Rochester, 1993b.

Polana, R. and R.C. Nelson, "Detecting Activities," Submitted for Publication, 1993.

IEEE Conf. on Computer Vision and Pattern Recognition, Urbana, IL, June 1992.

Pook, P.K. and D.H. Ballard, "Contextually dependent control strategies for manipulation," TR 416, Computer Science Dept., U. Rochester, April 1992a.

Pook, P.K. and D.H. Ballard, "Dexterous robotic gripper performs qualitative manipulation," *SPIE Robotics and Machine Perception Newsletter*, July 1992b.

Rimey, R.D. and C.M. Brown, "Controlling eye movements with hidden Markov models," *Int'l. J. of Computer Vision* 7, 1 (Special Issue), 47-65, 1991.

Rimey, R.D. and C.M. Brown, "Studying control of selective perception using T-World and TEA," to appear, *Proc., DARPA Image Understanding Workshop*, 1993.

Rimey, R.D. and C.M. Brown, "Task-specific utility in a general Bayes net vision system," *IEEE Conf. on Computer Vision and Pattern Recognition*, 142-147, Champaign, IL, June 1992a.

Rimey, R.D. and C.M. Brown, "Where to look next using a Bayes net: Incorporating geometric relations," *Proc., 2nd Eur. Conf. on Computer Vision*, 542-550, S. Margherita Ligure, Italy, May 1992b.

Schneider, J.G. and C.M. Brown, "Robot skill learning and the effects of basis function choice," TR 437, Computer Science Dept., U. Rochester, September 1992.

Soong, J. and C.M. Brown, "Inverse kinematics and gaze stabilization for the Rochester robot head," TR 394, Computer Science Dept., U. Rochester, August 1991;

Swain, M.J., L.E. Wixson, and D.H. Ballard, "Object identification and search: Animate vision alternatives to image interpretation," in P. Dario, G. Sandini, and P. Aebischer (Eds.). *Robots and Biological Systems: Towards a New Bionics?* Springer Verlag, NATO ASI Series, 1992.

Weems, C., C.M. Brown, J. Webb, T. Poggio, and J. Kender, "Parallel processing in the DARPA Strategic Computing Vision program," *IEEE Expert* 6, 5, 23-38, October 1991.

Wixson, L.E., "Looking Near One Object for Another," *Proc., SPIE, Intelligent Robots and Computer Vision XI: Algorithms, Techniques, and Active Vision* (Boston, MA), Vol 1825, D.P. Casasent, Ed., 159-167, November 1992.

Wixson, L.E. and D.H. Ballard, "Learning efficient sensing sequences for object search," *Proc., AAAI Fall Symp. on Sensory Aspects of Robotic Intelligence*, Asilomar, CA, November 1991.

Image Understanding Research at GE

J.L. Mundy*

Box 8

G.E. Corporate Research and Development
Schenectady, NY 12309

Abstract

Recent progress in image understanding research at GE is described. The focus of GE's program in IU is on the application of geometric constraint models and geometric invariants to the recognition and representation of objects, as well as the development of object-oriented software environments to support IU research and applications.

1 Overview

1.1 An Emphasis on Geometry

Image understanding research and applications at GE are centered around geometric descriptions and geometric reasoning for representing and recognizing objects. We have developed this geometric theme over the past decade with emphasis on object recognition and associated approaches for representing objects to facilitate recognition.

1.2 Constraint-Based Modeling

The conventional approaches to object recognition have been based on fixed polyhedral models or models with fixed relationships between components. We have been developing a system for representing broad categories of objects by defining an object in terms of geometric relations, or constraints. Any specific object of the class is considered to be a solution of the constraint system. In most cases, there is a continuum of solutions so a specific object is selected which satisfies the constraints as well as optimality criteria derived from image features.

A key application of our constraint-based modeling system is to reduce the manual effort in the construction of RADIUS¹ site models. Many building shapes can be represented by a generic set of constraints and the specific geometry of a building can be recovered by fitting the constraint model to various image views of the building. Recent progress on this application is described in section 1.3.2. A dual-use application of the constraint modeling system is automated interpretation of X-ray

images for the detection of flaws in jet engine turbine blades. This application is described elsewhere in these proceedings [Noble and Mundy, 1993].

1.3 Geometric Invariants

A significant body of results on the construction of geometric invariants to projective and affine transformations has been developed over the past three years [Mundy and Zisserman, 1992]. An invariant is any property of a geometric configuration which is unaffected by viewpoint. In current model-based vision approaches, it is necessary to test each object in the library since the specific properties of the object can only be exploited for discrimination after model pose is determined. When objects are described in terms of invariants the resulting properties can be used to index large model libraries. We have shown that invariant indexing leads to object recognition cost which grows slowly with the size of the model library. This efficient indexing property has been exploited in three application tasks.

1.3.1 Automatic Target Recognition

A key problem in automatic target identification is efficient indexing of targets from image features. This indexing must be done from a rather sparse and low-resolution image segmentation. It is often the case that the resolution is too low to permit robust indexing and recognition on geometric properties alone. We have recently developed a concept of integrated geometric and thermal invariants in conjunction with the Target Recognition Technology Branch at Wright Labs(AARA)² The idea is to derive a set of features which are invariant, not only to the geometric variations due to viewpoint, but also to the thermal variations due to environmental and target operational conditions. The benefits of invariant indexing can be applied to target detection and classification, even where resolution is limited, since attributes of the thermal intensity data are taken into account. The graphs in Figure 1 shows the values of several geometric-thermal invariants computed from a time series of IR images of a tank. The variation is quite small compared with the absolute temperature variation of individual tank components.

*The research reported here is funded in part by DARPA Contract #MDA972-91-C-0053

¹Research and Development for Image Understanding Systems, a joint project by DARPA and ORD.

²The key contributors at AARA are V. Velten, L. Westerkamp and M. Gauder. Substantial contributions have also been made by D. Forsyth of the University of Iowa.

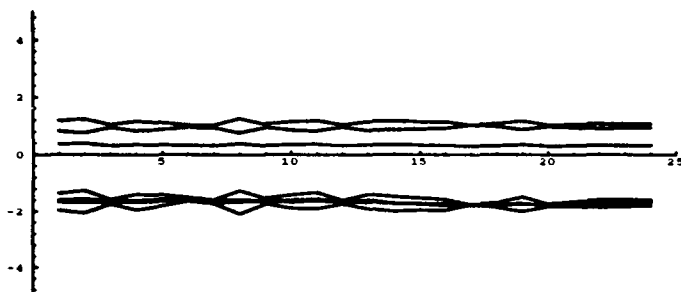


Figure 1: Ratio's of spatial-thermal integrals, as a function of time, for the carriage region of a tank. Note that four of the ratio's are effectively constant, and that the others vary only slightly over time.

1.3.2 Site Model Registration

The main RADIUS assumption is that each site will be described by a geometric site model which describes such features as: the site perimeter, lines of communication, functional areas and 3D building structure. This site model provides a context for IU tools to assist the image analyst in tasks such as asset accounting, change detection and scientific analysis. A central task for IU in the RADIUS program is the registration of existing site models to new reconnaissance imagery. In our approach, we represent features from the site model in terms of geometric invariants and then use these invariants to recognize key features at the site, such as buildings, roads and shorelines. This approach is described in more detail in section 3.1.

1.3.3 3D Reconstruction From Uncalibrated Cameras

Recently, Richard Hartley of GE and Olivier Faugeras of INRIA independently proved that a structure in 3D space can be reconstructed up to within a 3D projective transformation from two or more uncalibrated camera views. For many vision tasks such as object recognition, it is not necessary to determine the Euclidean structure of the object since the 3D object can be characterized by its projective invariants. If desired, the structure can be transformed to an appropriate Euclidean coordinate frame if a sufficient number of constraints, such as distances and orientations are known a priori about the object. This observation has lead to a new approach to acquiring and registering models to image data. Of pri-

mary importance is the ability to achieve most IU image analysis tasks without knowledge of the camera parameters or viewing conditions and without requiring ground control points [Hartley, 1993b].

1.4 Object-Oriented Design

In addition to our research and applications of object recognition and modeling techniques, we are involved in a number of projects for the application of object-oriented design to the development of research and application environments.

1.4.1 RCDE

Part of the DARPA sponsored RADIUS project, is to develop a common software environment(RCDE) to facilitate the exchange of results and to provide a platform for the demonstration of algorithms which are targeted at the intelligence exploitation of aerial images. The Cartographic Modeling Environment(CME) developed at SRI International by Lynn Quam, has been extended and interfaced to C++ by GE's Management and Data Systems Operation with support from GE-CRD to become the RCDE [Mundy *et al.*, 1992b]. Extensive documentation for operation and programming of the RCDE has also been developed. Currently a number of early versions of the RCDE are under evaluation and testing at RADIUS contractor beta sites. The RCDE is currently being enhanced by SRI and CRD under a task called THREAD which provides a narrow architecture for carrying out Model Supported Exploitation(MSE) using the RCDE. The key IU algorithmic steps in THREAD architecture are:

- Edge segmentation using a modified Canny edge detector and line segments extracted from the resulting pixel chains using maximum curvature.
- Line segment grouping based on endpoint proximity and collinearity.
- Site model registration using affine model matching based on vertex-pairs as well as affine and projective invariants.
- Linear feature extraction using image correlation.

Other elements of the THREAD architecture are related to camera modeling, interface to a database, and an analyst interface. This architecture provides a useful set of examples for using and integrating new applications with the RCDE.

1.4.2 IUE

A major DARPA project has been initiated to provide a standard software environment for carrying out image understanding research and applications. The Image Understanding Environment or IUE has been specified by a committee of senior IU researcher and is based on an object-oriented representation of the major data structures and operations used in IU algorithms. An overview of the IUE appears elsewhere in this proceedings [Mundy and Committee, 1993b]. The goal is that the IUE will become a standard of exchange of IU data and also provide common interfaces so that new algorithms and other support code can be freely exchanged between research

institutions. Recently effort at CRD has been focussed on the development and prototyping of a data exchange file format. It is envisioned that such a standard will be of considerable immediate benefit [Mundy and Committee, 1993a].

2 Constraint-Modeling

Our underlying approach to constraint modeling is the observation that all geometric constraints can be represented in terms of algebraic equations. A constraint-model is thus a system of polynomial equations and specific model instances correspond to root of the polynomial system. The final solution represents a best least squares fit with respect to both the constraint equations and the projection of corresponding image features. In several earlier papers we have provided details of the representation and have shown results on a number of solution techniques [Nguyen *et al.*, 1990, Nguyen *et al.*, 1991, Mundy, 1992].

The focus of this work over the last year has been the application of these techniques to RADIUS sites using the test data sets developed by ORD. In addition we have extended the use of constraints to curved 2D primitives which has enabled the representation of complex turbine blade geometries for automatic visual inspection [Noble and Mundy, 1993]. This application represents a nice instance of dual-use of government-funded technology.

2.1 RADIUS Site Models

There is now available imagery to support the analysis and site model construction for two sites, 1) aerial photography of Fort Hood and 2) an industrial area model-board. We have been applied the constraint model system to the representation of road networks as well as 3D building geometry. A first example is illustrated at the top of Figure 2 which shows a constraint model of an L-shaped building.

The operator roughly sketches the building and selects a few correspondences at key vertices as appropriate. The constraint system is then solved to maintain the correspondences while at the same time minimizing the error with respect to the image feature locations. The left image in Figure 2 shows the initial model as sketched. The final solution is shown in the right image. The process assumes the existence of a camera model for a single image of the scene. In this example a camera model was calculated from ground control points supplied with the RADIUS model-board.

A second 3D example is also shown in Figure 2. Here we constructed some 3D constraint primitives and used them to model a composite structure on the RADIUS model board. Again, the procedure was to roughly position the model in the vicinity of the actual building and then establish a few correspondences between the vertices of the constraint primitive and the image. The initial placement of the model primitives can be quite loose. For example one of the primitives is about 45° away from the true orientation. Note that the primitives do not have to be very close to the final shape and size in order to reach convergence. The only

Example	# Const.	Time	Error
Top Ex.	57	8 sec	.02 pixels
Bottom Ex.	141	66.8 sec	.58 pixels

Table 1: The performance of the constraint modeling system for the two examples. The computational complexity for the solver used in these experiments is N^3 where N is the number of constraints. The times shown are for a SPARC 2.

requirement is that the system of constraints established for the model is consistent with the final building shape.

The current version of the modeling system is written in C++ and the computational performance is characterized in Table 1.

3 Geometric Invariants

An invariant is a property of a set of geometric forms which does not change with viewpoint. Our premise is that invariants offer a sound framework for the representation of objects leading to efficient recognition algorithms. For the past few years we have worked jointly with Oxford University to develop and apply geometric invariants to the problem of object recognition³. A joint workshop between DARPA and ESPRIT, "Applications in Computer Vision," was held in Reykjavik, Iceland in April, 1991. The workshop brought together the leading researchers in invariant theory and applications. A collection of the papers from the workshop have been published by MIT Press [Mundy and Zisserman, 1992]. Also in November 1992, a very successful seminar on invariants was held at Wright Lab under the sponsorship of the Target Recognition Technology Branch and AFOSR. The two day seminar provided a tutorial on invariants presented by J. Mundy and D. Kapur⁴ as well as a session quasi-invariants presented by Tom Binford⁵. The seminar was attended by about 30 participants from government labs, ATR contractors and universities.

Below are some of the highlights of our recent results in the application of invariants.

3.1 Site Model Registration

Experiments have been carried out to test the performance of invariants on the RADIUS task of site model registration. The approach exploits the use of projective and affine planar invariants to locate major building features in images. After these features are recognized the full site model can be aligned with the image using standard camera resectioning software.

First we review the basic invariants used in the experiments.

³The individuals from Oxford University involved in the collaboration are A. Zisserman, D. Forsyth (now at the University of Iowa) and C. Rothwell

⁴State University of New York at Albany

⁵Stanford

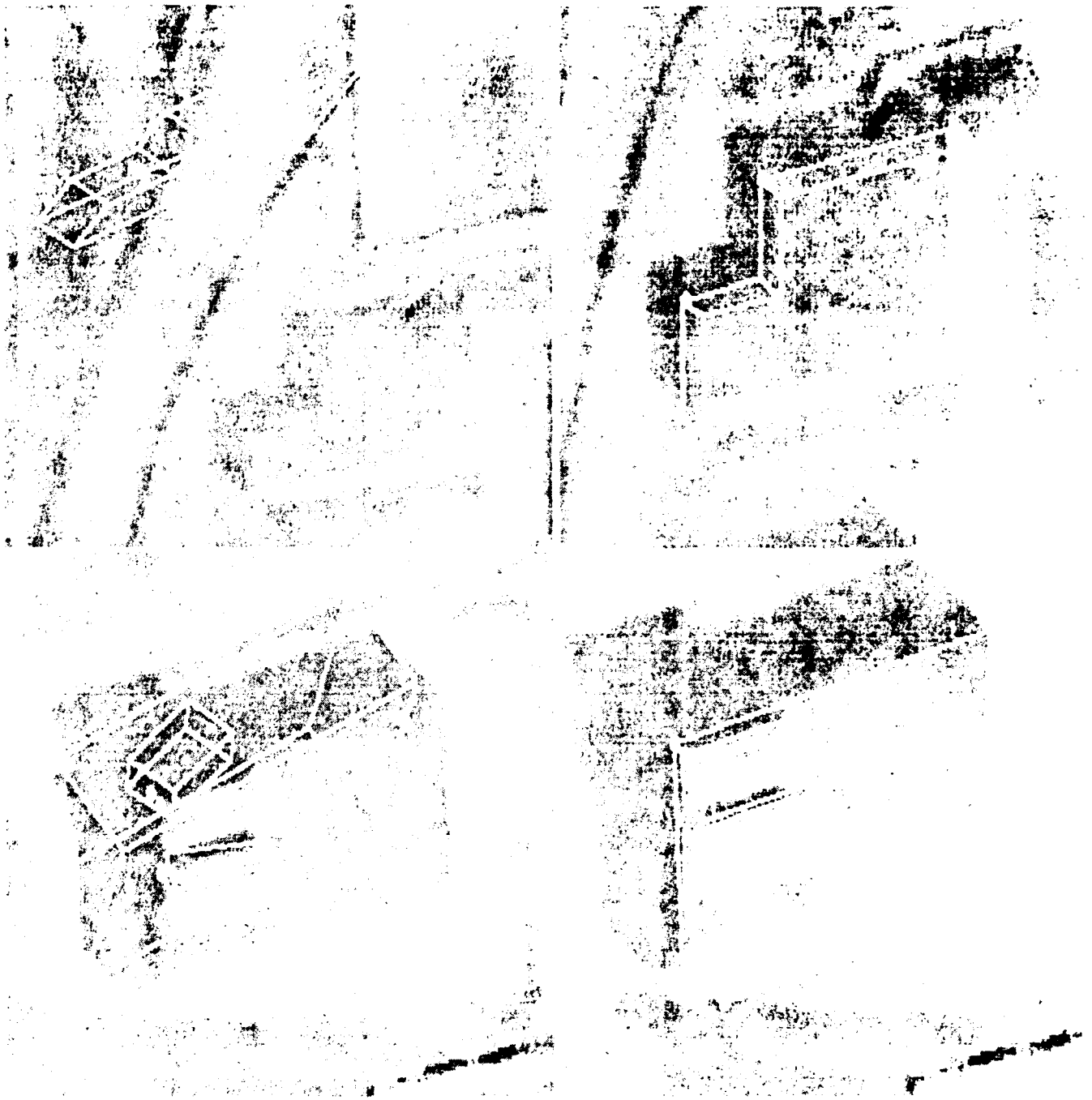


Figure 2: An example of constraint model building construction. The initial configuration of the model is shown in the image on the left. The final solution is on the right. Note that the constraints supply information about the shape of the structures so that only a few image feature point correspondences are needed to fix the shape.

Invariant 1: Five Coplanar Lines.

Given five coplanar homogeneous lines $l_i, i \in \{1, \dots, 5\}$, two projective invariants are defined:

$$I_{1a} = \frac{|M_{431}||M_{521}|}{|M_{421}||M_{531}|}$$

$$I_{1b} = \frac{|M_{421}||M_{532}|}{|M_{432}||M_{521}|}$$

where $M_{ijk} = (l_i, l_j, l_k)$ and $|M|$ is the determinant of M . Should any triple of lines become concurrent the first invariant is undefined. This singular case is common for polygons where alternate sides are parallel. In these cases we can only use the second invariant as a shape descriptor. Using the duality of points and lines the invariants can also be defined for five coplanar points, i.e. $M_{ijk} = (p_i, p_j, p_k)$.

Invariant 2: Two Points and Two Lines

A single projective invariant can also be derived from two points and two lines. The invariant is given by a ratio of various combinations of the algebraic distance from each point to each line, as follows.

$$I_2 = \frac{(l_1^t \cdot p_1)(l_2^t \cdot p_2)}{(l_1^t \cdot p_2)(l_2^t \cdot p_1)}$$

Note that each point and line appear the same number of times in the numerator and denominator so the projective scale factors cancel.

Experiments

An example of the use of feature matching based on projective invariants is shown in Figure 3. A set of invariants are used to describe the "L" shaped buildings at Fort Hood. The projective invariants are based on lines and points extracted from an edge segmentation of the building. Then other buildings of the same type in the same image or new images of the site are found by invariant indexing. Table 2 shows the value of typical invariants of each type within the same image and between images. The matches for each building are indicated by overlaying the segmentation edgels of the model building on each detected instance. In our experiments, building (1,1) was taken as the model. Neither projective and affine planar transformations account exactly for the actual RADIUS image characteristics. However, over a small field of view, such as a building, both of these models provide reasonable indexing power. That is, the variance of the invariant keys is small compared to the difference between invariant values of distinct object classes.

4 Basic Research Results

GE-CRD, in the context of a number of university collaborations, has continued to advance the basic foundations of object recognition, based on invariants. A key issue in geometric invariant research is the computation of invariants for 3D structures both from a single view and multiple views.

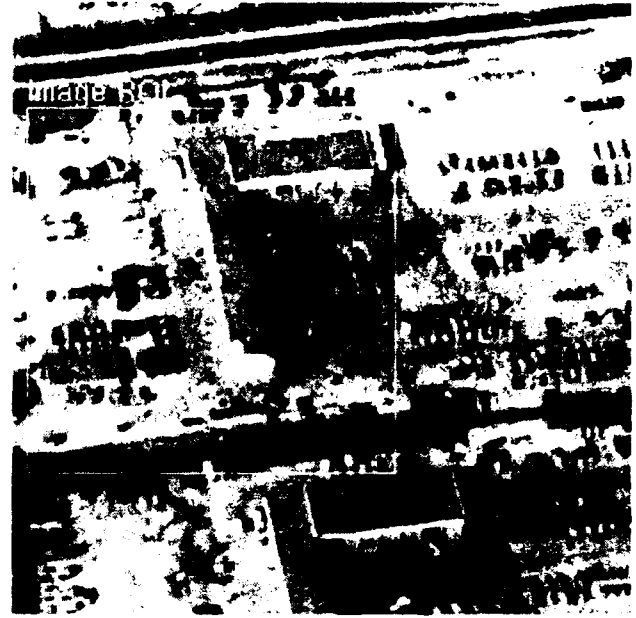


Figure 3: An example of feature matching using a combination of invariants. The match was based on features within the indicated ROI.

Building	I_1	I_2
11	0.770	0.212
12	0.782	0.208
13	0.781	0.221
21	0.776	0.236
22	0.782	0.240
23	0.782	0.230

Table 2: The value of various invariants for the "L" shaped building in the Fort Hood RADIUS images. The building label B_{ij} refers to building j in image i , so one can compare the variation in invariants between copies of same building design in the same image and across images. Only one five-line invariant is available because three or more of the building edges are typically parallel, a degenerate case.

4.1 Invariants for 3D Structures From a Single View

In general, for a single view, there are no invariants for a generic 3D structure. However, if one makes an assumption about the general class of a 3D object it is possible to establish a framework of invariants which can be reliably computed from a single view. Examples under development are:

Rotational Symmetry

A substantial body of results have been obtained for this case. It is possible to recover the axis of symmetry of a rotationally symmetric object from a single view and to use distinct points on the axis to compute invariant indices, based on the cross ratio. It is also the case that in a single view, the symmetrically corresponding occluding boundaries are within a plane projectivity of

each other. This fact enables the construction of a continuous invariant signature along the boundaries which can be used to refine the axis parameters. This work is described in more detail elsewhere in these proceedings [Liu *et al.*, 1993].

4.2 Polyhedral Invariants

It can also be shown that invariants can be constructed for general polyhedral surfaces where the faces have four or more vertices. This work is based on the earlier theory of Sugihara [Sugihara, 1986] which analyzed the degrees of freedom and correctness of image projections of polyhedra. We have extended his theory to permit the construction of projective invariants from uncalibrated camera views of polyhedral solids [Rothwell *et al.*, 1993]. The restriction to polyhedra is not very severe since a polyhedral "cage" can be invariantly constructed around a curved object surface with vertices established at points of high curvature.

4.3 Multiple Views

Recent work has established that invariants can be constructed for arbitrary 3D structures from two or more views. This work builds on previous developments by Barrett [Mundy *et al.*, 1992a]. It is possible to recover the epipolar structure of multiple image views from feature correspondences between views in terms of the essential matrix, Q . Once this matrix is available, the 3D geometry of an object can be constructed up to a projectivity of space. It is thus natural to describe objects in terms of their 3D projective invariants. For example, four lines in space define two projective invariants and six points define three projective invariants. The latter case is easy to see, since five points define a projective coordinate frame and the three coordinates of the remaining point, in this frame, are invariant.

Two invariants can be obtained from two views of a set of six points, with four points coplanar and two points not on the plane. This configuration has been exploited to effect a projective transfer of shapes between views without actually constructing the 3D geometry of the object [Demey *et al.*, 1992].

We have also extended our work on the extraction of structure from uncalibrated cameras to incorporate line features. An algorithm has been demonstrated which derives the epipolar structure from line features and a minimum of three uncalibrated camera views [Hartley, 1993a]. In addition, Hartley has shown that two, 3D projective invariants can be extracted from four lines. This work enables the recognition of objects using only line features [Hartley, 1993c].

References

- [Demey *et al.*, 1992] S. Demey, A. Zisserman, and P. Beardsley. Affine and projective structure from motion. In *Proc. BMVC92*, Springer Verlag, 1992.
- [Hartley, 1993a] R. I. Hartley. Camera calibration using line correspondences. In *Proc. DARPA Image Understanding Workshop*, 1993.
- [Hartley, 1993b] R. I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Proc. Second European Conference on Computer Vision*, Springer-Verlag, Vol. 588, 1993.
- [Hartley, 1993c] R. I. Hartley. Invariants of lines in space. In *Proc. DARPA Image Understanding Workshop*, 1993.
- [Liu *et al.*, 1993] J. Liu, J. L. Mundy, D. A. Forsyth, A. Zisserman, and C. Rothwell. Efficient recognition of rotationally symmetric surface and straight homogeneous generalized cylinders. In *Proc. DARPA Image Understanding Workshop*, 1993.
- [Mundy and Committee, 1993a] J. L. Mundy and IUE Committee. A exchange format for image understanding data. In *Proc. DARPA Image Understanding Workshop*, 1993.
- [Mundy and Committee, 1993b] J. L. Mundy and IUE Committee. The image understanding environment - overview. In *Proc. DARPA Image Understanding Workshop*, 1993.
- [Mundy and Zisserman, 1992] J.L. Mundy and A. Zisserman. *Geometric Invariance in Computer Vision*. MIT Press, Boston, MA, 1992.
- [Mundy *et al.*, 1992a] J. L. Mundy, P. M. Payton, M. H. Brill, and E. B. Barrett. Three dimensional model alignment without computing pose. In *Proc. 20th AIPR Workshop, SPIE Vol. 1623*, 1992.
- [Mundy *et al.*, 1992b] J. L. Mundy, R. Welty, L. Quam, T. Strat, W. Bremner, M. Horwedel, D. Hackett, and A. Hoogs. The radius common development environment. In *Proc. DARPA Image Understanding Workshop*, 1992.
- [Mundy, 1992] J. L. Mundy. Image understanding research at ge. In *Proc. DARPA Image Understanding Workshop*, 1992.
- [Nguyen *et al.*, 1990] V.-D. Nguyen, J. L. Mundy, and D. Kapur. Modeling polyhedra by constraints. In *Proc. DARPA Image Understanding Workshop*, 1990.
- [Nguyen *et al.*, 1991] V.-D. Nguyen, J. L. Mundy, and D. Kapur. Modeling generic polyhedral objects by constraints. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1991.
- [Noble and Mundy, 1993] J. A. Noble and J. L. Mundy. Constraint processing applied to industrial inspection and continuous process improvement. In *Proc. DARPA Image Understanding Workshop*, 1993.
- [Rothwell *et al.*, 1993] C. Rothwell, D. A. Forsyth, A. Zisserman, and J. L. Mundy. Extracting projective information from a single view of 3d point sets. In *Proc. 4rd International Conference on Computer Vision*, 1993.
- [Sugihara, 1986] Kokichi Sugihara. *Machine Interpretation of Line Drawings*. MIT Press, 1986.

A Conscious Observer: A Coordinated Effort in Computer Vision*

R. Bajcsy, D. Metaxas, M. Mintz & G. Provan

GRASP Laboratory, Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA

Abstract

According to the American Heritage Dictionary, the adjective "conscious" means having awareness of one's own existence and environment. In this paper we present some highlights of different aspects and/or competencies that a Conscious Observer must have. It has been said many times that some of these competencies are less conscious than others, or less context, task, and/or environment specific than others. We certainly agree with this, and accordingly our efforts can be categorized into at least two such cases or processing stages. The first stage can be characterized as physics-based understanding of reflectance (described in Section 1) and physics-based shape and motion modeling and estimation (described in Section 2). During the second stage, we process multisensory observations and/or multiple features or parameters using statistical decision rules that will result in either physical action (obstacle avoidance, as an example) or mental action. The mental action in this case is the indexing stage of recognition of objects. Section 3 describes

the sensory fusion process and the statistical decision theory that the sensory fusion process is based on, while Section 4 shows the problems and issues with representation, i.e., identifying the most statistically salient features for indexing into an object database to recognize categories of shapes. There are two common threads running through our various research projects. One is that since the image formation process is physics-based, it is just common sense to use physics-based models in order to perform the inverse. The other thread is the realization that the scenes and the measurements that we make are noisy, incomplete and vary due to environmental effects and hence, common sense again dictates that one must use multiple sensory measurements and multiple representations. The theory that models these processes in a most appropriate way seems to be statistical decision theory, which we are applying.

1 Physics-Based Preprocessing (R. Bajcsy)

During the last four years we have been engaged in understanding of the interaction of light (illumination) and surfaces. This understanding, which is based purely on physical principles, has led to the development of several algorithms that enable us to identify, locate and remove, if desirable, highlights, interreflections, shading and shadows [Funka-Lea, 1992:

*This research was supported in part by the following grants and contracts: Navy Grant N00014-92-J-1647, AFOSR Grants 88-0244, AFOSR 88-0296; Army/DAAL 03-89-C-0031PRI; NSF Grants CISE/CDA 88-22719, IRI 89-06770, and ASC 91 0813; NSF CISE/CDA-90-2253, NSF MSS-91-57156, NATO Grant No. 0224/85, Barrett Technology Inc., A.I. duPont Institute and Du Pont Corporation

Bajcsy et al., 1990; Lee and Bajcsy, 1992; Lee, 1991].

The motivation for this work comes from the hypothesis that highlights, interreflections, shading and shadows often create artifacts in partitioning of the scene that do not correspond to the true physical reality, and hence it is desirable to remove them prior to segmentation of the scene. The difficulty lies in the fact that the imaging sensor measures an integral of all the above components. The research question is: what principled constraints and/or other measurements can one bring to bear in order to be able to separate the components? Of course we are not the only group addressing this problem; currently there is a whole community working in this area [Boult and Wolff, 1991; Gershon et al., 1986; Healey and Binford, 1989; Klinker et al., 1990; Nayar et al., 1991; Shafer, 1985].

At the last Image Understanding workshop [Bajcsy, 1992] we reported the results describing identification of highlights from metallic objects and surfaces with varied albedo. In this section we shall concentrate on understanding shadows. Shadows result from the obstruction of light from a source of illumination. As such, shadows have two components: one spectral and one geometric. The spectral nature of a shadow derives from the characteristics of the light illuminating the shadow as compared to the additional light that would illuminate the same area if there was no obstruction. Hence, shadows reveal themselves as a spectral change in radiance due to a change in the local irradiance. The geometry of a shadow is determined by the nature of the illumination obstruction and the scene geometry. A light source may be only partially obstructed. In fact, for any non-point light source, the outer portion of a shadow results from the partial obstruction of the light source. This is the penumbra of the shadow, while the umbra is the part of the shadow where the light is completely obstructed.

First, we shall introduce our model of shadows without other reflectance effects. Let D be the amount of energy from the illumination source, measured at a given surface and let E be the integral of all other illumination sources in the en-

vironment. Let I represent the light measured by the camera from one point of view. Coefficient β from interval $[0, 1]$ indicates the amount of obstruction, where $\beta = 0$ corresponds to the umbra of the shadow. The value $\beta = 1$ corresponds to the surface directly lit. The images values I of the surface in and out of shadow are proportional to the linear relationship:

$$I \propto \beta D + E.$$

This relationship in turn is used to recover a single surface directly lit and in shadow as a single image region. It is also used as an aid in identifying the umbra and penumbra of a shadow. However, this equation will not help with discrimination between shadow penumbra and/or shading. It also may not apply for multiple light sources and varied albedos. For these cases, one needs more constraints! The additional constraints come from: (i) using a shadow casting probe; (ii) using spatial segmentation based on some homogeneity criterion; (iii) using geometric constraints on where shadows can be cast; or (iv) using active light cast into the scene. We enumerate the different cases in Table 1 below.

For brevity, we demonstrate our results on one example that uses color image segmentation. This algorithm is based on three ideas: (i) using line-like color models to take into account shadow candidate regions; (ii) dovetailing the processing between color-space and image-space; and (iii) looking for the best description of an image in terms of primitive models via region segmentation. The region growing process recovers uniform or linear functions in color space. Region growing is initiated from seed regions found from the highest peaks in the color histogram, or if no peaks are found, based on laying a grid of small regions on the image. The result of this process is shown in Figure 1.

2 A Physics-Based Framework for Shape and Nonrigid Motion Estimation (D. Metaxas)

2.1 Introduction

In the past couple of years we have dealt with the robust and efficient estimation of shape and nonrigid motion and addressed several related

Identifying Shadows

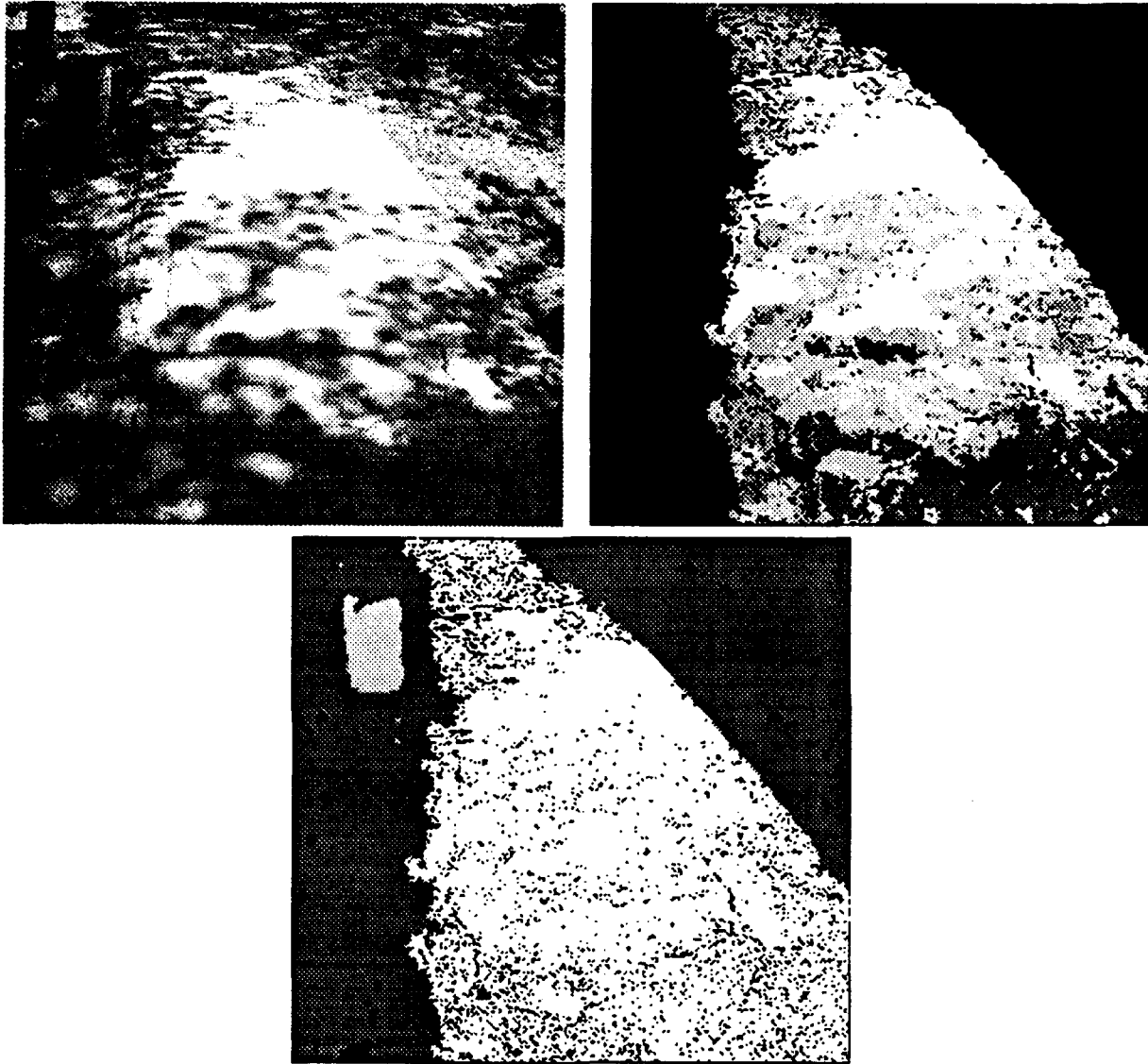


Figure 1: **Top Left:** An image of a road directly lit and in shadow courtesy of the Carnegie Mellon Navlab project. **Top Right:** A gray scale labeling of the umbra and penumbra of the shadows on the road as recovered by our color image segmentation for shadows. Shadow umbra is indicated by dark gray, penumbra by light gray, and the road directly lit by white. **Bottom:** The full color image segmentation of the original image aimed at recovering single materials directly lit and in shadow as single image regions. The different regions are indicated by different, suggestive colors. Black indicates that no region was found at that image position.

Table 1: Shadow Case Study. The columns indicate lighting conditions under which shadows may be cast. The rows indicate material properties within the scene and whether or not shadows include an umbra. A 'y' indicates that the case can be handled, while an 'n' indicates that the case cannot be handled in general by our system. The label *Multiple Albedos* indicates distinct constant albedos, while the label *Varying Albedos* indicates smoothly varying albedos and material properties. The column in which an observer's environment falls can be determined by examining a shadow actively cast by the observer.

* For this case, the umbra alone is represented as a linear color cluster, as opposed to the other cases our system can handle, where a single material both in and out of shadow is represented by a linear color cluster.

Material Changes	Umbra & Penumbra	Shadows			
		None	1 Light Source	2 Light Sources	More Than 2 Light Sources
One Albedo	U & P	y	y	y *	n
	P	y	y	n	n
Multiple Albedos	U & P	y	y	n	n
	P	y	y	n	n
Varying Albedos	U & P	n	n	n	n
	P	n	n	n	n

difficult problems. We have consequently developed a physics-based framework for 3D shape and nonrigid motion modeling which includes: (i) a new class of dynamic deformable primitives which combine global and local deformation parameters; (ii) a systematic approach based on Lagrangian dynamics and the finite element method to convert the geometric parameters of the primitives to dynamic degrees of freedom; (iii) the development of physics-based constraints between these deformable primitives that may be used to track the motions of complex articulated objects; (iv) a recursive technique for estimating shape and non-rigid motion from noise corrupted data based on applying Kalman filtering theory to our dynamic models; and (v) new applications to visual estimation. In what follows we elaborate on each of the above technical contributions, which have also been reported in [Terzopoulos and Metaxas, 1990; Metaxas and Terzopoulos, 1991a; Metaxas and Terzopoulos, 1991b; Metaxas and Terzopoulos, 1993].

2.2 New Deformable Primitives

We have created a new family of modeling primitives by developing a mathematical approach that allows the combination of global

and local deformations. Our primitives include global deformation parameters which represent the salient shape features of natural parts and local deformation parameters which capture shape details. More specifically, we have developed hybrid models whose underlying geometric structure allows the combination of parametric models (superquadrics, spheres, cylinders), parameterized global deformations (bends, tapers, twists, shears, etc.) and local spline free-form deformations. In this way, the descriptive power of our models is a superset of the descriptive power of locally deformable models [Terzopoulos and Witkin, 1988] and globally deformable models [Pentland and Horowitz, 1991; Witkin and Welch, 1990]. An important benefit of the global/local descriptive power of these models is that it can potentially satisfy the often conflicting requirements of shape reconstruction and shape recognition. The local degrees of freedom of deformable models allow the reconstruction of fine scale structure and the natural irregularities of real world data, while the global degrees of freedom capture the salient features of shape that are innate to natural parts and appropriate for matching against object prototypes.

2.3 Systematic Formulation of Dynamic Primitives

Through the application of Lagrangian mechanics, we have developed a method to systematically convert the geometric parameters of the solid primitive, the global (parameterized) and local (free-form) deformation parameters, and the six degrees of freedom of rigid-body motion into generalized coordinates or dynamic degrees of freedom. More precisely, our method applies generally across all well-posed geometric primitives and deformations, so long as their equations are differentiable. The resulting Lagrange equations of motion which describe the dynamic behavior of our models take the form

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{g}_q + \mathbf{f}_q,$$

where \mathbf{M} , \mathbf{D} , and \mathbf{K} are the mass, damping, and stiffness matrices, respectively, where \mathbf{g}_q are inertial forces arising from the dynamic coupling between the local and global degrees of freedom, and where $\mathbf{f}_q(u, t)$ are the generalized external forces (computed from the forces that the data exert on the model) associated with the degrees of freedom q of the model.

The distinguishing feature of our approach is that it combines the parameterized and free-form modeling paradigms within a single physical model. Thus our models exhibit correct mechanical behaviors and their various geometric parameters assume well-defined physical meanings in relation to prescribed mass distributions, elasticities, and energy dissipation rates. Furthermore, motivated by the requirements of real time vision applications we appropriately simplify the models and use simple numerical integration techniques to achieve real time or near real time simulation rates on available graphics workstations.

2.4 Physics-Based Constraints

To deal with constrained multipart objects such as articulated anthropomorphic bodies, we have developed an efficient technique to implement hard point-to-point constraints between deformable primitives. These constraints are never violated, regardless of the magnitude of the forces experienced by the parts. Attempting to approximate such constraints with sim-

ple, stiff springs leads to instability. In our approach [Metaxas, 1992], we compute the constraint forces using a stabilized Lagrange multiplier technique [Baumgarte, 1972].

2.5 Recursive Estimation

We also have exploited the constrained nonrigid motion synthesis capabilities of our models in order to estimate shape and motion from incomplete, noisy observations available sequentially over time. Applying continuous nonlinear Kalman filtering theory [Metaxas and Terzopoulos, 1991b; Metaxas and Terzopoulos, 1993], we have constructed a powerful new recursive estimator which employs the Lagrange equations of 3D nonrigid motion as a system model. We interpret the Kalman filter physically: the system model continually synthesizes nonrigid motions in response to generalized forces that arise from inconsistencies between its state variables and the incoming observations. The observation forces account formally for instantaneous uncertainties in the data. A Riccati procedure updates an error covariance matrix which transforms the forces in accordance with the system dynamics and the prior observation history. The transformed forces induce changes in the translational, rotational, and deformational state variables of the system model to reduce the inconsistencies. Thus the system model synthesizes nonstationary shape and motion estimates in response to the visual data.

2.6 Experiments

The following experiments demonstrate the application of the above described techniques to various data. Fig. 2 illustrates the fitting of a deformable superquadric to a monocular image of a pestle Fig. 2(a). The image is converted into a force field that acts on the model, deforming it such that it becomes consistent with the occluding boundary of the pestle in the image. Fig. 2(b) shows the initial state of the deformable superquadric displayed in wireframe projected onto the image. Fig. 2(c) shows an intermediate step in the fitting process as the image forces are deforming the model and Fig. 2(d) shows the final reconstructed model.

The following two experiments demonstrate the

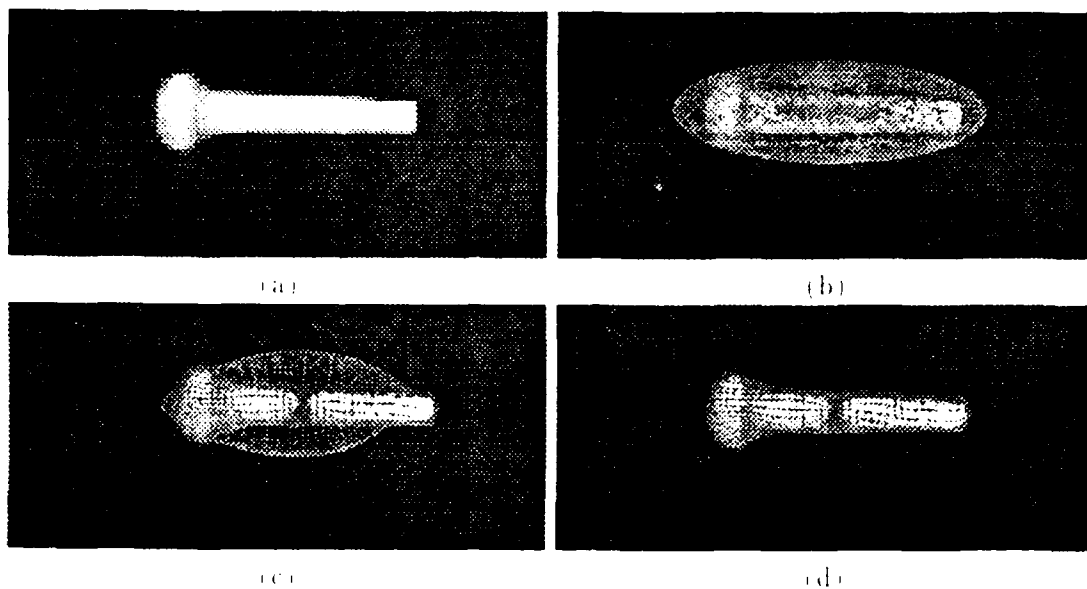


Figure 2: Fitting a deformable superquadric (b), (c), (d) to pestle image (a).



Figure 3: Tracking of raising and flexing human arm motion.

performance of our recursive shape and motion estimator. In the first experiment the estimator incorporates constrained deformable superquadrics as Kalman filter system models. The figure illustrates a model composed of five connected deformable superquadrics. The estimator is applied to biomechanical data collected by 3D position sensors applied to the arms of a human subject. Fig. 3(a) shows a view of the 3D data and the initial models. Fig. 3(b) shows an intermediate step of the fitting process driven by data forces from the first frame of the data sequence, while Figs. 3(c) and (d) show different views of the models fitted to the initial data. Figs. 3(e) and (f) show intermediate frames of the models tracking the nonrigid motion of the subject's flexing arms, while Figs. 3(g) and (h) show two views of the final position of the models.

In Fig. 4 we add uniform noise, by perturbing by $\pm 5\%$ (with randomly chosen sign) the noiseless value of the 123 motion range data point and we fit a deformable superquadric with 81 nodes. Figs. 4 (a) and (b) show two views of the range data and the initial model. Fig. 4(c) shows an intermediate step of the fitting process driven by data forces from the first frame of the motion sequence. Figs. 4(d) and (e) show the model fitted to the initial data, with visible tapering and bending global deformations. Fig. 4(f) shows an intermediate frame of the model tracking the nonrigid motion of the squash, while Figs. 4 (g) and (h) show the final position of the squash.

3 Multisensor Fusion (M. Mintz)

3.1 Introduction

The successful design and operation of autonomous or partially autonomous agents which are capable of traversing uncertain terrains requires the application of multiple sensors for tasks such as: reconnaissance, surveillance, and target acquisition and/or manipulation. In applications which include a teleoperation mode, there remains a serious need for local data reduction and decision-making to avoid the costly or impractical transmission of vast quantities of sensory data to a remote operator. There are several reasons to include multisensor fusion in a system design: (i) it allows the designer to

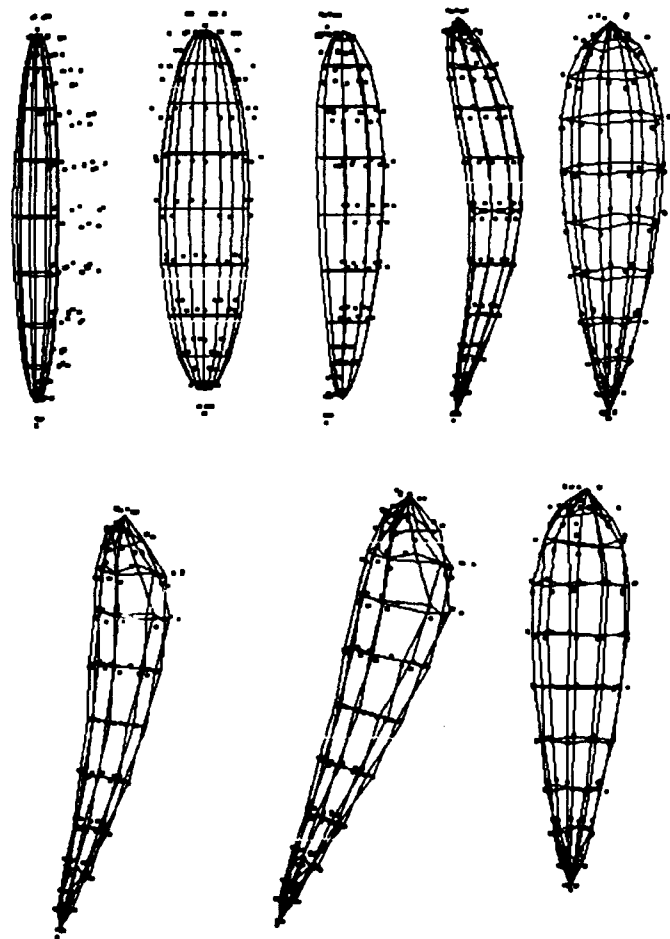


Figure 4: Tracking of fully deformable squash shaped object with noise.

combine intrinsically dissimilar data from several sensors to infer some property or properties of the environment, which no single sensor could otherwise obtain; and (ii) it allows the system designer to build a robust system by using partially redundant sources of noisy or otherwise uncertain information.

3.2 Sensor Fusion Research Issues

The following task-related issues arise in the design and operation of autonomous systems which employ multiple sensors: (i) the value of a sensor suite; (ii) the layout, positioning, and control of sensors (as agents); (iii) the marginal value of sensor information; the value of sensing-time versus some measure of error reduction, e.g., statistical efficiency; (iv) the role of sensor models, as well as *a priori* models of the environment; and (v) the calculus or calculi by which consistent sensor data are determined and combined.

3.3 Technical Rationale

An important role for active sensing is the surveillance and sensory exploration of environments that are characterized by significant *a priori* uncertainties. In addition to uncertainty in the environment, the sensors themselves exhibit noisy behavior. While good engineering practice can reduce certain noise components, it is impractical if not impossible to eliminate them completely. Thus, all sensor measurements are uncertain. However, sensor errors can be modeled statistically, using both physical theory and empirical data. In developing these models, one recognizes that a single distribution is usually an inadequate description of sensor noise behavior. It is much more realistic and much safer to identify an envelope or class of distributions, one of whose members could represent the actual statistical behavior of the given sensor. This use of an uncertainty class (or equivalently an envelope, set, or neighborhood) in distribution space protects the system user against the inevitable unpredictable changes that occur in sensor behavior. Reasons for uncertainty in statistical sensor models include: sporadic interference, drift due to aging, temperature variations, miscalibration, quantization, and other significant nonlinearities over

the dynamic range of the sensor.

3.4 Approach

Our approach to sensor fusion employs statistical decision theory to obtain: (i) a robust test of the hypothesis that data from different sensors are consistent; and (ii) a robust procedure for combining the data that pass this preliminary consistency test. Here, robustness refers to the statistical effectiveness of the decision rules when the probability distributions of the observation noise and the *a priori* position information associated with the individual sensors are uncertain.

We have developed a coherent decision-theoretic approach to robust multisensor fusion which provides the means to compute hard probabilistic confidence measures of data consistency and robustly combine consistent sensor data [Kamberova and Mintz, 1990; McKendall and Mintz, 1992]. Our approach allows the system designer to explicitly incorporate *a priori* information in the form of geometric constraints, and also make use of set-valued uncertainty class models which capture the noise behaviors of the various sensors. This work is particularly important because: (i) it allows the system designer to incorporate geometric constraints or information about the features or parameters of interest without requiring generally insupportable assumptions about *a priori* probabilities, e.g., the uniform distribution on the set; (ii) it allows the system designer to incorporate realistic sensor noise behavior in the analysis without requiring the very often insupportable "Gaussian hypotheses"; and (iii) the sensor noise distribution may be an element of a nonparametric set of distributions which are asymmetric, multimodal, heavy-tailed, and generally nonmonotone likelihood ratio.

Because our methodology easily allows for the accurate incorporation of geometric constraints, we are consequently able to address sensor fusion tasks in which both wide and narrow field-of-view sensors are employed. Specifically, we can make robust confidence set-based tests for correspondence between features at coarse and fine scales.

3.5 Current and Related Research

We have studied the theory and application of robust fixed-size confidence intervals as a methodology for robust multisensor fusion. This work has been delineated in [Kamberova and Mintz, 1990] and [McKendall and Mintz, 1992]. Currently, we are implementing a DARPA-funded multiagent hardware-software testbed for studying multisensor fusion, and multiagent communication and cooperation [Bajcsy et al., 1992]. This testbed is based on: (i) mobile robotic agents with multiple sensors and manipulatory capability; and (ii) mobile observer (sensory) agents.

Our sensor fusion studies focused initially on confidence intervals as opposed to the more general paradigm of confidence sets. The basic distinction here is between fusing data characterized by an uncertain scalar parameter versus fusing data characterized by an uncertain vector parameter, of known dimension. While the confidence set paradigm is more widely applicable, we initially chose to address the confidence interval paradigm, since we were simultaneously interested in addressing the issues of: (i) robustness to nonparametric uncertainty in the sampling distribution; and (ii) decision procedures for small sample sizes. Our research on optimal and robust fixed-size confidence intervals has appeared in [Zeytinoglu and Mintz, 1984; Zeytinoglu and Mintz, 1988].

We have also investigated the multivariate (confidence set) paradigm. The delineation of optimal confidence sets with fixed geometry is a very challenging problem when: (i) the *a priori* knowledge of the uncertain parameter vector is *not* modeled by a Cartesian product of intervals (a hyper-rectangle); and/or (ii) the noise components in the multivariate observations are *not* statistically independent. Although it may be difficult to obtain optimal fixed-geometry confidence sets, we have obtained some very promising approximation techniques. These approximation techniques provide: (i) statistically efficient fixed-size hyper-rectangular confidence sets for decision models with hyper-ellipsoidal parameter sets; and (ii) tight upper and lower bounds to the optimal confidence coefficients in the presence of both Gaussian and non-Gaussian sampling distributions. We have

shown, through numerous examples of these applications, that the risks of the approximating procedures are within 0.5% of optimal. These approximation techniques have been reported in [Kamberova et al., 1992].

Further, we generalize these previous decision-theoretic results in two important directions:

(1) We obtain optimal and near-optimal fixed-size confidence intervals for restricted location parameters for sampling distributions which do not possess monotone likelihood ratio. Examples of this sort of distribution are the Cauchy distribution, and Gaussian distributions with heavy-tailed contamination. We derive a class of nonmonotone almost equalizer rules for this decision problem. We establish that rules in this class achieve near-minimax risk. In particular, in the case of the Cauchy sampling distribution we show, by example, that the risk is within 0.3% of optimal. We also establish that very general shift and scale mixtures of Gaussian distributions have optimal procedures with a very simple monotone form. Since these Gaussian mixtures are generally not monotone likelihood ratio, this suggests that a critical factor which determines the need to consider nonmonotone rules is the tail-behavior of the sampling distribution. We obtain results which delineate this connection. These results [Kamberova and Mintz, 1993] extend the work on monotone procedures [Zeytinoglu and Mintz, 1984; Zeytinoglu and Mintz, 1988].

(2) We obtain minimax rules for restricted location parameters under symmetric multilevel bowl-shaped loss for symmetric sampling distributions with monotone likelihood ratio. Multilevel bowl-shaped loss functions are obtained by a convex combination of n zero-one loss functions with given width parameters. Sufficient conditions for minimax Bayes rules are derived. These conditions are easy to check numerically. The minimax rules possess the following structure: the rules are continuous (or piecewise-continuous), piecewise-linear functions with alternating segments of zero and unit slope. These rules are simple to compute numerically. Further, we show: (i) how to approximate arbitrary symmetric bowl-shaped loss functions using multilevel approximants; and (ii) how to

obtain accurate approximations to the minimax rules for decision problems with symmetric bowl-shaped loss functions and restricted parameter spaces. An outcome of this approximation study is the result that the minimum Fisher information prior (\cos^2) defines a bounding envelope for the least favorable prior distributions when the scale parameter of the sampling distribution tends to zero. These results [Kennedy and Mintz, 1993] extend the work on zero-one loss [Zeytinoglu and Mintz, 1984; Zeytinoglu and Mintz, 1988], and on the role of the minimum Fisher information prior [Bickel, 1981].

4 Object Recognition (G. Provan)

4.1 Introduction

Over the past year work has been done in generalizing existing object recognition capabilities, building upon previous work done in the GRASP Lab on recovering superquadric representations for multi-part objects from dense range data [Solina, 1987; Gupta and Bajcsy, 1990]. The primary contributions are principled solutions to the difficult problems of superquadric part classification and model indexing, and *object class* recognition. This project consists of two stages: (i) object database creation, and (ii) recognition.

4.2 Object Database Creation

Each object is represented as a set of superquadric parts, and each superquadric part in turn is represented by a superquadric parameter vector \mathbf{m} . In creating an object database, first, we cluster together similar model parts to create a reasonable number of prototypical part classes. Second, we statistically analyze the parameter-sets to identify the statistically most significant subset of parameters \mathbf{m}' which distinguish objects (or object classes) from one another. This is achieved by selecting a small but highly diagnostic subset of the parameters or by combining the original parameters to yield a small number of new, more diagnostic features, such as height-to-width ratio, squareness, etc. For any given domain, such distinguishing keys may be computed using statistical techniques such as principal components analysis.

The most salient sub-vector \mathbf{m}' is used as a smaller-dimension initial index into the database during object recognition. This improves upon the *ad hoc* nature of the feature/parameter vectors used in most recognition systems.

Additional benefits of reduced-dimensionality vectors include greater recognition robustness (since many elements of the vector are often just "noise", and the remaining elements have more accurate estimated mean and covariance matrices used in classification), faster search (fewer variables to match/search), smaller databases, and more efficient overall object recognition.

4.3 The Recognition Stage

We start with range data from a single rigid multi-part object. We are currently using the segmentation algorithms developed by Solina and Gupta, although we hope to incorporate the alternative techniques developed by Metaxas [Metaxas, 1992]. Once an initial superquadric fit has been done, each of the superquadric parts recovered from the input is paired with the best matching prototypical part class using precomputed class statistics. The indexing keys used are the most distinguishing parameters or parameter-clusters.

This matching process produces a probability associated with the data/model compatibility for a set of object subparts. A formal evidential approach is then used to compute the probability of this collection of subparts being a multi-part object.

This feature-selection approach to indexing has been tested on a domain of simple concave kitchen utensils (e.g. bowls, cups, pots, etc.). Linear and quadratic classifiers were trained and tested on a collection of 64 representative objects and a set of 3 parameters were identified as accounting for 98% of the variance, a significant reduction in the dimensionality of the feature space. These three parameters were then used for indexing using dense range data from representative domain objects.

Future work involves a full implementation of the subpart evidential combination scheme, and active vision routines to cope with poor segmentations or low match probability. If recognition

is not possible (e.g. due to the viewpoint leading to an inability to uniquely distinguish the object), this active vision approach should provide the capability to use a new range image (from a different viewpoint) to improve the segmentation and/or re-run the recognition algorithm.

References

- [Bajcsy et al., 1990] R. Bajcsy, S. W. Lee and A. Leonardis. "Color image segmentation with detection of highlights and local illumination induced by inter-reflections", *Proc. 10th International Conf. on Pattern Recognition*, Atlantic City, NJ, June, 1990.
- [Bajcsy, 1992] R. Bajcsy. "An Active Observer", *Proc. of the DARPA Image Understanding Workshop*, Morgan Kaufmann, San Mateo, pp. 137-147, January, 1992.
- [Bajcsy et al., 1992] R. Bajcsy, V. Kumar, M. Mintz, R. P. Paul and X. Yun. "A small-team architecture for multiagent robotic systems", *Proc. of the Workshop on Intelligent Robotic Systems: Active and Purposive Systems*, SPIE, Boston, November, 1992.
- [Baumgarte, 1972] J. Baumgarte. "Stabilization of Constraints and Integrals of Motion in Dynamical Systems", *Computer Methods in Applied Mechanics and Engineering*, 1:1-16, 1972.
- [Bickel, 1981] P. J. Bickel. "Minimax estimation of the mean of a normal distribution when the parameter space is restricted", *Ann. Statist.*, 9(6):1301-1309, November, 1981.
- [Boult and Wolff, 1991] T. E. Boult and L. B. Wolff. "Physically based edge labeling", *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 656-662, Maui, Hawaii, 1991.
- [Funka-Lea, 1992] G. Funka-Lea. *A proposal concerning the analysis of shadows in images by an active observer*, Department of Computer and Information Science, University of Pennsylvania, Technical Report MS-CIS-92-78, GRASP LAB 335, October, 1991.
- [Gershon et al., 1986] R. Gershon, A. D. Jepson and J.K. Tsotsos. "Ambient illumination and the determination of material changes", *Journal of the Optical Society of America*, 3(10):1700-1707, October, 1986.
- [Gupta and Bajcsy, 1990] A. Gupta and R. Bajcsy. "Part Description and Segmentation Using Contour, Surface and Volumetric Primitives", *SPIE Conf.*, pp. 203-214, 1990.
- [Healey and Binford, 1989] G. H. Healy and T. O. Binford. "Using color for geometry-insensitive segmentation", *Journal of the Optical Society of America*, 6, 1989.
- [Kamberova and Mintz, 1990] G. Kamberova and M. Mintz. "Robust multi-sensor fusion: A decision-theoretic approach", *Proc. of the DARPA Image Understanding Workshop*, Morgan Kaufmann, San Mateo, pp. 867-873, September, 1990.
- [Kamberova and Mintz, 1993] G. Kamberova and M. Mintz. "Fixed size confidence intervals for non-mlr location problems", *Meeting of the Institute of Mathematical Statistics*, IMS, Philadelphia, March, 1993.
- [Kamberova et al., 1992] G. Kamberova, R. McKendall and M. Mintz. *Multisensor Data Fusion Based on Fixed-Geometry Confidence Sets*, Department of Computer and Information Science, University of Pennsylvania, Technical Report MS-CIS-92-31, GRASP LAB 312, April, 1992.
- [Kennedy and Mintz, 1993] R. Kennedy and M. Mintz. "Minimax estimation under multilevel loss", *Meeting of the Institute of Mathematical Statistics*, IMS, Philadelphia, March, 1993.
- [Klinker et al., 1990] G. J. Klinker, S. A. Shafer and T. Kanade. "A physical approach to color image understanding", *International Journal of Computer Vision*, 4, 1990.

- [Lee, 1991] S. W. Lee. *Understanding of Surface Reflections in Computer Vision by Color and Multiple Views*, Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania, 1991.
- [Lee and Bajcsy, 1992] S. W. Lee and R. Bajcsy. "Detection of specularities using color and multiple views", *Proc. of ECCV-92*, Italy, May, 1992.
- [McKendall and Mintz, 1992] R. McKendall and M. Mintz. "Data fusion techniques using robust statistics", *Data fusion in robotics and machine intelligence*, eds. M. A. Abidi and R. C. Gonzalez, Academic Press, New York, pp. 211-243, 1992.
- [Metaxas, 1992] D. Metaxas. *Physics-Based Modeling of Nonrigid Objects for Vision and Graphics*, Ph.D. thesis, Department of Computer Science, University of Toronto, 1992.
- [Metaxas and Terzopoulos, 1991a] D. Metaxas and D. Terzopoulos. "Constrained Deformable Superquadrics and Nonrigid Motion Tracking", *Proc. IEEE Computer Vision and Pattern Recognition Conference (CVPR '91)*, Hawaii, pp. 337-343, June 1991.
- [Metaxas and Terzopoulos, 1991b] D. Metaxas and D. Terzopoulos. "Recursive Estimation of Nonrigid Shape and Motion", *Proc. IEEE Motion Workshop*, Princeton, NJ, pp. 306-311, October 1991.
- [Metaxas and Terzopoulos, 1993] D. Metaxas and D. Terzopoulos. "Shape and Nonrigid Motion Estimation Through Physics-Based Synthesis", *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1993, in press.
- [Nayar et al., 1991] S. K. Nayar, K. Ikeuchi and T. Kanade. "Surface reflection: Physical and geometrical perspective", *IEEE Trans. PAMI*, 13:611-634, 1991.
- [Pentland and Horowitz, 1991] A. Pentland and B. Horowitz. "Recovery of Non-rigid Motion and Structure", *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7):730-742, 1991.
- [Shafer, 1985] S. A. Shafer. "Using color to separate reflection components", *COLOR Research and Application*, 10:210-218, 1985.
- [Solina, 1987] F. Solina. *Shape Recovery and Segmentation with Deformable Part Models*, Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania, 1987.
- [Terzopoulos and Metaxas, 1990] D. Terzopoulos and D. Metaxas. "Dynamic 3D Models with Local and Global Deformations: Deformable Superquadrics", *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7):703-714, 1991. See also *Proc. Third International Conference on Computer Vision (ICCV'90)*, Osaka, Japan, pp. 606-615, December, 1990.
- [Terzopoulos and Witkin, 1988] D. Terzopoulos and A. Witkin. "Physically Based Models with Rigid and Deformable Components", *IEEE Computer Graphics and Applications*, 8(6):41-51, 1988.
- [Witkin and Welch, 1990] A. Witkin and W. Welch. "Fast Animation and Control of Nonrigid Structures", *Computer Graphics (Proc. SIGGRAPH)*, 24(4):243-252, 1990.
- [Zeytinoglu and Mintz, 1984] M. Zeytinoglu and M. Mintz. "Optimal fixed size confidence procedures for a restricted parameter space", *Ann. Statist.*, 12(3):945-957, September, 1984.
- [Zeytinoglu and Mintz, 1988] M. Zeytinoglu and M. Mintz. "Robust fixed size confidence procedures for a restricted parameter space", *Ann. Statist.*, 16(3):1241-1253, September, 1988.

IU AT UI: AN OVERVIEW OF RESEARCH DURING 1991-92

Narendra Ahuja and Thomas Huang

Beckman Institute and Coordinated Science Laboratory

University of Illinois

405 North Mathews Avenue

Urbana, Illinois 61801

Abstract

This paper presents an overview of the research in image understanding (IU) at the University of Illinois (UI) conducted during 1991-92. During this period, our work has been in five areas: integration for three-dimensional vision, motion analysis, analysis guided synthesis, representation and navigation, and learning and recognition. Work in each of these areas is reviewed.

1 Introduction

We review here the research progress we have made since [33]. This includes the progress (Secs. 2-5) on previously ongoing projects in the four areas reported in [33], as well as work in a new area (Sec. 6). The first area (Sec. 2) is concerned with integration of multiple image cues in performing image interpretation. These cues capture different aspects of the three-dimensional (3D) scene structure, and their integrated analysis leads to a more robust inference about the scene characteristics than possible from individual cues. The second area (Sec. 3) reports our work on interpretation of image sequences showing dynamic scenes. Here we consider the problems of detecting feature correspondences and estimating the 3D motion parameters and surface structure from feature correspondences, in a sequence of images showing motion which is rigid or nonrigid, and motion specific or relatively general. The third area (Sec. 4) is concerned with analysis guided synthesis of scenes which we introduced in [33, 24]. Here the goal is to synthesize images for depiction of 3D characteristics of the scene, using attributes recovered during interpretation or artificial attributes. The use of image attributes identified during 3D recovery takes advantage

of 3D analysis for perceptually realistic synthesis. The fourth area (Sec. 5) is concerned with different components of an evolving 3D representation and navigation system which has the goal of autonomously acquiring, maintaining and using 3D information about the environment. We have begun work in the area of learning object recognition strategies (Sec. 6). The goal here is to learn to automatically perform extraction and recognition of a class of objects from examples of such recognition. Representative projects in each of these areas are summarized in the following sections. To keep the paper brief, we have minimized discussion of and references to relevant work done by others. Such discussion and references are available in the cited and other listed publications.

2 Integration

Our goal in this area is to perform image interpretation such that the interpretation simultaneously satisfies a range of constraints imposed by the image structure and the model of the scene. To do this, we use different computational processes each of which carries complementary or redundant information derived from different image cues. Image interpretation is the result of a cooperative computation that resolves conflicts and ambiguities arising from the individual processes. We have presented several examples of the integration approach in previous IU workshops [31, 32, 33]. Here we summarize some recent work on integration.

2.1 Integrated Active Stereo

The goal of our continuing work on active stereo is surface estimation from stereo images of large scenes having large depth ranges, where it is necessary to aim cameras in different directions to fixate at different objects and to construct the global surface map of the scene from small patches. The first stage of this work involves surface reconstruction of a single object, having no depth discontinuities. It performs integration of camera vergence, focus, aperture, stereo and calibration processes. The second stage allows scenes containing arbitrarily placed and arbitrary size objects. In this stage, a part of the visual field that has not yet been fixated but

*Parts of this research were supported by the Defense Advanced Research Projects Agency and the National Science Foundation grant IRI-8902728, National Science Foundation under grant IRI-89-08255 Army Research Office Advanced Construction Technology Center under grant DAAL 03-87-K-0006, Joint Services Electronics Program under grant N00014-90-J-1270, and the State of Illinois Department of Commerce and Community Affairs under grant 90-103.

has appeared as the peripheral visual field, helps determine the next fixation point and provides coarse (inaccurate) structural information, to be refined following future fixations. Details of these stages can be found in [31, 32, 33, 1] and the references cited therein. We have now started a study to analytically compare the performances of the binocular cues of stereo and vergence, and the monocular cue of focus for estimating scene surfaces [2]. For ease of analysis, this is done by considering the estimation of range of a scene point, thus excluding the changes in the range values that would result from the use of surface smoothness constraint during surface fitting. The performance of the individual cues is evaluated as a function of errors in their respective parameters. Two types of errors, called systematic and random errors, are identified for each of the range estimation methods. The effects of random, quantization errors are expressed in terms of the mean and variance of the resulting depth error. Analytical expressions for the effects of systematic, calibration errors on estimation using each cue are also obtained. Further, we have developed a simplified approach to modeling the spatial quantization error in axial stereo vision systems with rectangular pixel geometry [56]. The need for simplification arises due to the lack of symmetry and spatial invariance of the image disparity. Numerical simulations show that the modeling accuracy is adequate for practical purposes, and points to the underlying complexity of the true error distributions. Such performance evaluation of the individual cues is useful for identifying the imaging parameters whose control would be most effective in improving the range estimate, and for devising strategies to integrate the use of the cues in order to combine their strengths and to overcome their individual limitations.

We have made further progress [57, 58] on the multi-component blurring (MCB) model presented in [33]. Unlike previous blurring models, this model can capture emergent image details that occur due to depth-discontinuities in the scene. It also means that MCB effects do not obey the maximum principle in scale-space, and thus blurring can create spurious details. We have simulated Pentland's depth-from-blur algorithm in normal and MCB blurring cases (MCB cases are not tractable by Pentland's method). Previously we had suggested that human perception of blur and depth could be enhanced by the presence of MCB effects. We now have more supporting data from independent psychophysical studies concerning an interesting phenomenon in human blur perception that has some correlation to the MCB effects. We have done extensive experiments on the MCB effects in real images.

2.2 Integrating Camera Panning and Depth from Focus using a Novel Camera

In integrated active stereo, the cameras must scan large visual fields by fixating different parts, and at each fix-

ation, integrate the information from multiple cues. We have now developed a new approach which integrates panning, focusing, and range estimation. To experiment with this, we have developed a novel camera system whose image plane tilt with respect to the optical axis is controllable, and the two common mechanical operations of focusing and panning are replaced by panning alone. Consequently, range estimation takes place at the speed of panning. Thus, imaging geometry and optics are exploited to eliminate explicit sequential computation. Since the camera implements a range from focus approach, the resulting estimates have the advantages and disadvantages of any such approach. For details of this work, see [3] in these proceedings.

2.3 Integrating Shape Estimation from Stereo and Shading

We have developed an approach to the integration of shape information provided by stereo with that provided by shading for estimating surface maps [4]. Such integration is facilitated by the use of color images which are more easily segmented than gray level images. The integrated system is able to accurately obtain depth estimates under a wider range of conditions than either stereo alone or shape from shading alone. Specifically, integrating stereo and shape from shading has several advantages. Stereo algorithms cannot accurately determine depth for large featureless regions. Errors in surface shape are locally large but are independently distributed so that global errors tend to be no larger than local errors. On the other hand, shape from shading tends to be locally accurate but can cause large global errors, especially if the boundary conditions are not well known. In an integrated system, shape from shading can use small features that would not be resolved by a stereo system. At the same time, stereo can provide initial conditions, boundary conditions and stability to the iterative solution process used in shape from shading. It can also provide the initial estimate of the depth map which is required for estimating the light source direction. We have developed an approach to estimation of the light source distribution to facilitate better interpretation of shading, and thereby more robust integration of stereo and shading [5]. Shape from shading algorithms are limited in their applicability by the assumption of overly simplistic models of the complex light source distributions that occur in the real world. We have obtained a more complete representation in which the lighting model makes use of multiple fixed point sources located at infinity. Methods of estimating the model parameters are developed and a method of estimating the surface shape given the source distribution is presented. The shape estimation algorithm using multiple sources is based on a new, single source algorithm which is an improvement over existing shape from shading algorithms. The source distribution algorithm and the generalized algorithm for shape from shading and stereo have been applied to images of real

and artificial scenes.

2.4 Integrating Target Tracking and Coarse-to-fine Surface Estimation

We have begun to investigate the problem of tracking a target object in a complex visual environment under binocular viewing, and simultaneously generating an evolving surface map [7, 8]. A corner feature detector is applied to each image to locate point features. To initiate the tracking process, we use correlation across stereo images to match the feature points both in space and in time, thus producing partial 3D trajectories without prior knowledge of target motion. The unknown object motion is assumed to be piecewise smooth. To capture such object motion, we use the Autoregressive Moving Average (ARMA) model to fit the 3D motion trajectory. The ARMA model is initialized by each partial 3D trajectory obtained, and then the model is used to extend the trajectory to predict future image plane positions of feature points. The stereo correspondences between feature points are established by comparing the predicted and the actual image locations of the feature points. The 3D positions of the feature points are computed from their image plane location. The trajectories are extended, the ARMA models are updated, and the camera configurations are modified to achieve tracking. The surface maps of the target object and the background are constructed by fitting the 3D feature points with smooth surface patches. Since the target object is maintained in fixation, the accuracy of the environment range map depends on the location of the target relative to other surfaces. This yields target tracking and simultaneous generation of surface map.

This as well as our integrated active stereo algorithms [31, 32, 33] have been implemented on the University of Illinois Active Vision System described in [6]. The system employs two high-resolution cameras for image acquisition and is capable of automatically directing movements of the cameras so that camera positioning and image acquisition are tightly coupled with visual processing. The system was designed and developed in 1987 as a research tool, largely based on off-the-shelf components. A central workstation controls imaging parameters, which include five degrees of freedom for camera positioning (tilt, pan, translation, and independent vergence) and six degrees of freedom for the control of two motorized lenses (focus, aperture, and zoom). In [6], we have described the hardware of the system, the imaging model, the calibration method employed and some of the system software. A second version of this system has recently been constructed and placed on an autonomous vehicle.

2.5 Integrating Region and Border Extraction for Image Structure Detection

We have begun work on developing a new transform to extract the edge contours and skeletons of image re-

gions at multiple scales. We have considered the application of the transform to detecting edge structure. The transform is motivated by the observation that linear processing based approaches, such as convolution and matching, have the fundamental deficiency of using a priori models of edge geometry. The proposed transform avoids this limitation by letting the structure "emerge," bottom-up, from interactions among pixels, in analogy with statistical mechanics and particle physics. The transform involves global computations on pairs of pixels followed by vector integration of the results, rather than the commonly used scalar, local, linear processing. An attraction force field is computed over the image. Pixels belonging to the same region are mutually attracted whereas those across edges repel each other. Scale is an integral parameter of the force computation. The resulting groupings of pixels represent multiscale image structure. The properties desired in multiscale edge detection are given, and it is theoretically and experimentally shown that the transform possesses these properties. Along with their contours, the transform also extracts skeletons of multiscale regions. Preliminary experimental results with synthetic and real images demonstrate the above properties of the transform. Details of this work are given in a separate paper in these proceedings [9]. Our recent work on integration of Gestalt constraints for dot pattern grouping can be found in [28].

2.6 Computational Models of Integration

One formalism for modeling the process of integration using dynamical systems is presented in [30, 10]. According to this model, visual processing is performed in parallel at each location in an image by multiple, relatively simple dynamical systems. Multiple vision computations are unified by many interacting dynamical systems. For example, features may be identified with limit sets of a multi-attractor system. The position of the feature can be obtained by mapping the profile of, say, the Laplacian-of-Gaussian of the image onto a limit cycle attractor where phase along the limit cycle corresponds to relative image position. Similarly, velocity information can be recovered by mapping the temporal derivative of the Laplacian-of-Gaussian operator onto a different component of the dynamics. The design and analysis of a three-dimensional nonlinear dynamical system, in which the position and motion profiles of an intensity edge are mapped onto a two-dimensional submanifold of the model dynamics, are discussed in [10]. This demonstrates a simple form of integration by embedding two inputs within a single system. An array of such dynamical systems can be used for detecting spatio-temporal trajectories in the image where the analog nature of the dynamical systems ensures real time performance.

3 Motion Analysis

The long-range goal of our research in this area is the understanding of dynamic scenes. We have made progress in three major subareas: finding feature correspondences in image sequence, determining rigid motion parameters and surface structure from the correspondences, and analyzing nonrigid motion.

3.1 Detecting Feature Correspondences

Detecting feature correspondences is difficult due to a wide variety of 3D structural discontinuities and occlusions that occur in real world scenes. A major part of our work on this problem is concerned with matching point features. Our objective is to integrate identification of feature correspondences with segmentation and motion and structure estimation. We have begun to develop an approach to detecting and segmenting feature trajectories in image sequences having multiple moving objects that may have temporal discontinuities in their motion. One common type of motion discontinuity is in motion direction, e.g. when an object undergoes collision. The object surfaces are assumed to contain point features that are automatically detected as the images are acquired. The process of feature tracking begins when the second frame is acquired and the feature points detected. With two frames, feature tracking amounts to finding two-view correspondences. As more frames become available, tracking becomes equivalent to extending the trajectories already determined, and matching with feature points in the next frame. In both cases, appropriate constraints are used to compute costs that are associated with every candidate match. These constraints involve image plane similarity in the arrangement of neighbors around the points, smoothness in the 3-D motion of objects and smoothness in the image plane motion of the features. The costs computed using the above constraints are merged together to obtain a single cost. This cost is incorporated into an energy function along with the uniqueness constraints, and this energy function is minimized using a Hopfield network. The problem of removing wrong correspondences that may result from local minima is solved using another Hopfield-like network. Details of this work are presented in a separate paper in these proceedings [11]. In another effort, we have considered feature extraction and matching as special cases of the more general problem of signal detection [12]. Our early work on two view matching appears in [35].

3.2 Rigid Motion and Structure from Image Sequences

Our work in this area is concerned with estimating motion and structure of a scene from feature correspondences. We are interested in segmentation of the sequence into distinctly moving objects, as well as in the estimation of the motion and structure of each object. As stated earlier, our objective has been to integrate de-

tection of correspondences, segmentation, and motion and structure estimation.

We have developed sufficient conditions for double or unique solution of the problem of motion and structure estimation of a rigid surface from pairs of monocular images [27]. These conditions further the understanding of the uniqueness problem of rigid motion estimation. We show that 5 correspondences of noncolinear points that do not lie on a special type of quadratic curve, called Maybank Curve, in the image plane suffice to determine a pure rotation uniquely, and 6 correspondences of points that do not correspond to space points lying on a Maybank Quadric suffice to determine a motion with nonzero translation uniquely. We show that each Maybank quadric can sustain at most two physically acceptable motion solutions and surface interpretations, provided that a sufficient number of correspondences are present. In particular, we show that in the plane motion case, 6 correspondences of points that do not lie on a quadratic curve in the image plane only admit the true motion and structure and their duals as solutions. We discuss how noise affects the uniqueness of solution and present a nonlinear algorithm for estimation of motion parameters.

We have developed algorithms for estimating motion and structure parameters from long monocular image sequences by using the most appropriate of a set of long sequence motion models [13, 14, 15]. We first present a new two-view motion algorithm and then extend it to long sequence motion analysis. The two-view motion algorithm requires generally 6 pairs of point correspondences to give unique solution of the motion parameters. However, when the points used for correspondences lie on a Maybank Quadric, the algorithm requires 7 pairs of point correspondences to give all possible double solutions. Object-centered motion representations and models of motion described by up to the second order polynomials are analyzed. Two long sequence algorithms are presented, one using interframe matches, and the other using point trajectories. The long sequence algorithms automatically find the proper model that applies to an image sequence and gives the globally optimal solution for the motion and structure parameters under the chosen model. Since the algorithm does not involve structure parameters, it contains fewer unknowns than usual which makes it more efficient and robust. Experimental results with several real image sequences showing different motions demonstrate the performance of the algorithm.

We have shown that the motion and structure of rigidly moving objects can be completely determined from two monocular image sequences using only temporal matches [55]. Three aspects of this scheme are useful: (1) since stereo matching is not necessary, two cameras can view totally different parts of the rigid scene; (2) as temporal disparity is usually significantly smaller than stereo disparity, matching needs only to deal with relatively small disparities; and (3) the re-

coverable scene structure is defined by the union of the fields of view of two cameras instead of the intersection, and so is much larger than that of a conventional stereo setup. Experiments with synthesized data and real world images demonstrate the feasibility of this scheme.

We have developed necessary and sufficient conditions for determining shape and motion to within a mirror uncertainty from orthographic projections of any number of point trajectories over any number of views [16]. We prove that there are always two sets of solutions of shape and motion under orthographic projection: if shape S is a solution, so is its mirror image S which is symmetric to S about the image plane. The necessary and sufficient conditions for determining the two sets are associated with the rank of the measurement matrix W . We prove that if the rank of W is 3, then a necessary and sufficient condition is to be satisfied to determine the solution to within a mirror uncertainty. If the condition is not satisfied, then infinitely many solutions result. If the rank of W is 2 and the image points in at least one view are not colinear in the image plane, then there are two possibilities: either the motion is around the optical axis or the 3-D points all lie on the same plane. In the first case, the motion can be determined uniquely but the shape is not determined. In the second case, a necessary and sufficient condition is to be satisfied and at least 3 point trajectories over at least 3 views are needed to determine the shape in each view to within a mirror uncertainty. If the rank of W is 2 or 1 and the points in each view are colinear in the image plane, then the three dimensional motion problem reduces to a two dimensional motion problem. In this case, a necessary and sufficient condition needs to be satisfied to determine the shape and motion to within a mirror uncertainty. All proofs are constructive and can be used to build a completely linear algorithm for estimating shape and motion from point trajectories.

Our factorization based approach [33] to integrated motion and structure estimation from orthographic image sequences of arbitrary length and containing arbitrary numbers of features per frame is presented in [29, 17]. Our other recent work on motion and structure estimation is reported in [34, 39, 36, 21]

3.3 Nonrigid Motion

We have continued work on interpretation of image sequences showing nonrigidly moving objects such as fluids, deformable objects (human face), and articulated objects (human body).

Fluid motion is a major focus of nonrigid motion research. Unlike rigid body or elastic body motion analysis, fluid motion analysis is based on the vortex structures in the velocity fields of fluid. Particle tracking velocimetry is one way to obtain velocity vectors of a turbulence on random positions experimentally. To further analyze the motion of the fluid, we need to interpolate for the velocity vectors at regular grid points.

Previously, we developed a physically-constrained robust interpolation method. The velocity fields were considered as nonrandom functions, and multivariate reciprocal quadratics were chosen as the interpolants for their nice mathematical properties. We now model the velocity field as a vector-valued multidimensional random process where the particle particles are approximated by a Poisson sampling process [59]. Based on this model, the velocity interpolation problem becomes that of interpolation of a multidimensional random process. While considering the special characteristics of fluid flow, we extended previous work on random process interpolation from scalar-valued one-dimensional to vector-valued multidimensional for homogeneous turbulence. An optimum linear filter which minimize the mean squares error is derived for this interpolation problem. The mean squares errors of this interpolation are analyzed against the fluid property as well as the particle density. Compared with previous interpolation methods, the interpolants in this work are the correlation functions of turbulent flows which have clear physical meaning and could be obtained by theoretical analysis or by experiments.

Human face modeling is an important problem in applications such as videophone, teleconferencing and person identification. We have developed a method to obtain a standard 3D wire-frame model of a person's face from only the front and two side views [60]. The generic face model we have used consists of a set of connected triangular meshes. For each view of the face, the model is fit to the face. Each view supplies sufficient depth information to modify the model parameters. The model fit is then compared with the real 3D data of the face to obtain a measure of relative error. For this, we first fit the face model to the real 3D data by least-squares matching of several key points, and then find the corresponding points on the 3D data according to the nodes of the face model to compute the error. The results of our experiments show that this method can generate a realistic 3D face model for a person, which can be further used for facial motion analysis and expression synthesis. Such 3D model-based coding differs from conventional waveform coding in that it makes use of 3D properties of the objects. It uses an explicit 3D model for the object, encodes images based on computer vision techniques and recovers the original images with computer graphics methods. Since it only transmits several analysis parameters, an extremely low bit rate of image transmission can be achieved. We are also continuing our study of the visual motion of human ambulatory patterns.

4 Analysis Guided Synthesis

In [33, 24], we introduced a framework for image synthesis using the information extracted during 3D interpretation. The objective is identification and depiction of image attributes such that the display effectively communicates the 3D scene structure as seen by an observer

in relative motion to the scene. We have continued and extended this work [25]. There are two major aspects of this research. First, it introduces the notion that the cues that contribute the most to three-dimensional interpretation are also the ones that would yield the most realistic synthesis, thus suggesting an approach to analysis guided compression. Second, it presents an approach to recovering 3D motion and structure parameters from multiple cues present in a monocular image sequence, such as point features, optical flow, regions, lines, texture gradient and vanishing line. The use of line features has been recently incorporated into our implementation of the approach. For concreteness, this work focuses on flight image sequences of a planar, textured surface. The integration of information in the diverse cues is carried out using optimization. In our recent work, the requirement that motion be smooth is no longer imposed and the vanishing line in each frame is now automatically recognized from the detected lines by using the motion estimates from two successive views. For reliable estimation, a sequential batch method is used to compute motion and structure. For synthesis, real and/or artificial attributes are shown as a monocular sequence or as a binocular (stereo) sequence thus further highlighting the recovered motion and structure parameters. Experiments have been conducted with two image sequences, one digitized from a commercially available videotape as reported in [24] and a new sequence acquired from a laserdisc. This second sequence is more challenging to our algorithm since the images contain partially or completely occluded vanishing lines and there is reflection of the ground in the bottom of the airplane. The quality of the images is somewhat better than that of the VHS tape used as the source of the first sequence, resulting in better estimates. Image compression ratios achieved for these sequences are 502 and 367 per frame. However, since the motion and structure parameters do not change significantly with each new frame, the contents of a frame can be estimated from the structure computed from other nearby frames, and the motion and structure parameters. Consequently, only 1 out of every n frames may be used (say for transmission in a communication scenario), thus increasing the compression ratios achieved to $502n$ and $367n$, respectively. A stereo display of the results has also been developed on Silicon Graphics workstation, which can be viewed using stereo glasses. The display sequence appears very similar to the original sequence in informal, monocular as well as binocular viewing.

The use of domain specific (model based) cues in integrated analysis results in identification of model components. In the first sequence, this is demonstrated by the use and identification of the vanishing line. In the second sequence, the constraint that runway edges are parallel to each other as well as to the direction of translation results in identification of these edges. Such identification of model components (in addition to more general scene characteristics) can be viewed as "analysis

guided recognition" [26]. Figure 1 shows experimental results for the second sequence in which the runway is recognized and then enhanced in the synthesized images.

5 Representation and Navigation

Our goals in this area continue to be two. First, we are interested in efficient computation of representations of the shape information such as acquired by three-dimensional estimation algorithms described in previous sections. Second, we are interested in using the scene representations for path planning. Our work on both these problems uses potential field as computational tool. Details of our initial work on potential field based approach to path planning are presented in [20]. In [33] we summarized our potential field based approach for efficient derivation of the medial axis transform and the generalized cylinder representations of a two-dimensional region [18]. Further, also reviewed how the skeletons of rigid moving objects are used in conjunction with potential field based representation of free space for solving path planning problems using closed form expressions for repulsive force. We have now extended this work to path planning for planar robot arms. Algorithms are developed for obtaining arm configurations of minimum Newtonian potential by constraining the skeleton of the robot arm according to the given path topology [19]. We have also surveyed the work on gross motion planning, i.e., motion planning without contacts between robots and objects, for point robots, rigid robots and manipulators in stationary, time-varying, constrained, and movable-object environments [22]. It reviews numerous research results on motion planning reported during 1985-1992 by researchers in various disciplines such as robotics, artificial intelligence and computational geometry. It presents a taxonomy of motion planning problems and a classification of various motion planning approaches which is used to structure the survey. Each type of motion planning problem is explained and its complexity is described. Relevant algorithms are explained briefly and their performances are compared with other algorithms. Future research directions are suggested for each motion planning problem.

6 Learning and Recognition

We have begun work on learning recognition of object classes, including the segmentation or extraction of object area [37, 38]. Learning involves automatically estimating the criteria for segmentation and classification in terms of multiscale structural constructs, called concepts. These constructs or concepts are dynamically formed in terms of basic image features such as edges from a large number of recognition examples.

A framework called Cresceptron is introduced for automatic algorithm design through learning of concepts and rules needed for the development of algorithms.

thus deviating from the traditional mode in which humans specify the rules which comprise a vision algorithm. With Cresceptron, the designers need only to provide a good structure for learning, but they are relieved of most design details. Cresceptron is tested on the task of visual recognition: recognizing 3-D general objects from 2-D photographic images of natural scenes, and segmenting the recognized objects from the cluttered image background. Cresceptron uses a hierarchical structure to grow networks automatically, adaptively and incrementally through learning. Each neural plane in the network hierarchy gets automatically associated with a different type of concept, the concepts being detected automatically, and the network grows by creating new nodes and connections which memorize the new concepts and their context. Cresceptron makes it possible to generalize training examples to other perceptually equivalent items. Segmentation and recognition are simultaneous. No foreground extraction is necessary, which is achieved by backtracking the response of the network down the hierarchy to the image parts contributing to recognition. Several types of network structures have been developed, and their properties are studied in terms of knowledge recallability, positional invariance, generalization power, discrimination power and space complexity. Experiments with a variety of real-world images have been performed to demonstrate the feasibility of learning in Cresceptron.

References

- [1] S. Das and N. Ahuja, Integrating Coarse-to-Fine Image Acquisition and Surface Estimation, *Sadhna*, 1993, to appear.
- [2] S. Das and N. Ahuja, Performance Analysis of Focus, Vergence and Stereo as Depth Cues for Active Vision, submitted.
- [3] A. Krishnan and N. Ahuja, Range Estimation from Focus Using a Nonfrontal Imaging Camera, *these proceedings*.
- [4] Darrell R. Hougen and Narendra Ahuja, Integration of Stereo and Shape from Shading Using Color, *Proc. Second International Conference on Automation, Robotics and Computer Vision*, Vol. 1, pp. CV-6.6.1-CV-6.6.5, Sept. 1992.
- [5] Darrell R. Hougen and Narendra Ahuja, Estimation of the Light Source Distribution and its Use in Shape Recovery from Stereo and Shading, *Fourth ICCV*, Berlin, Germany, 1993, to appear.
- [6] A. L. Abbott and N. Ahuja, The University of Illinois Active Vision System, *Proceedings: SPIE Conference on Applications of Artificial Intelligence XI*, Boston, Nov. 1992, pp. 757-768.
- [7] Z. Hong and N. Ahuja, Target Tracking from Binocular Image Sequence Using the Autoregressive Moving Average Model, *Proc. IAPR Workshop on Machine Vision Applications*, Tokyo, Japan, Dec. 1992, 317-320.
- [8] Z. Hong and N. Ahuja, Target Tracking and Cumulative Depth Map Generation from Binocular Image Sequences, *Proc. 3rd International Conference on Intelligence Autonomous Systems*, Pittsburgh, PA, Feb. 1993, to appear.
- [9] N. Ahuja, A Transform for Detection of Multiscale Image Structure, *these proceedings*.
- [10] Edward J. Altman, Bifurcation Analysis of Chua's Circuit with Applications for Low-Level Visual Sensing, *Journal of Circuits, Systems and Computers*, Vol. 3, No. 1, 1993, to appear.
- [11] S. Thirumalai, Detection and Segmentation of Feature Trajectories in Multiple, Discontinuous Motion Image Sequences, *these proceedings*.
- [12] X. Hu and N. Ahuja, Feature Extraction and Matching as Signal Detection, *Proc. SPIE Conference Applications of Artificial Intelligence XI: Machine Vision and Robotics*, 1993, to appear.
- [13] X. Hu and N. Ahuja, Estimating Motion of Constant Acceleration from Image Sequences, *Proc. 11th ICPR*, Hague, Netherlands, August 1992.
- [14] X. Hu and N. Ahuja, Long Image Sequence Motion Analysis Using Polynomial Motion Models, *Proc. IAPR Workshop on Machine Vision Applications*, Tokyo, Japan, Dec. 1992.
- [15] X. Hu and N. Ahuja, Motion and Structure Estimation using Long Sequence Motion Models, *Journal of Image and Vision Computing*, to appear.
- [16] X. Hu and N. Ahuja, Necessary and Sufficient Conditions for Determining Shape and Motion from Orthographic Projection, submitted.
- [17] C. Debrunner and N. Ahuja, Segmentation and Factorization-based Motion and Structure Estimation for Long Image Sequences, *IEEE Trans. PAMI*, to appear.
- [18] J. Chuang and N. Ahuja, Skeletonization using a generalized potential field model, *The Eighth Israeli Conference on Artificial Intelligence and Computer Vision*, Tel-Aviv, Israel, Dec 1991.
- [19] J. Chuang and N. Ahuja, Robot Arm Path Planning Using the Newtonian Potential, in *Proceedings of International Conference on Control and Robotics*, Vancouver, Canada, 14-17 August 1992.
- [20] Y. Hwang and N. Ahuja, A Potential Field Approach to Path Planning, *IEEE Trans. on Robotics and Automation*, Vol. 18, No. 1, 1992, 23-32.
- [21] X. Hu and N. Ahuja, Motion Estimation Under Orthographic Projection, *IEEE Trans. on Robotics and Automation*, December 1991, 848-853.
- [22] Y. Hwang and N. Ahuja, Gross Motion Planning - A Survey, *ACM Computing Surveys*, 219-292, Fall 1992.

- [23] A. N. Choudhary, J. H. Patel, and N. Ahuja, A Hierarchical and Partitionable Architecture for Computer Vision, *IEEE Trans. on Parallel and Distributed Systems*, to appear.
- [24] S. Sull and N. Ahuja, Integrated 3D recovery and Visualization of Flight Image Sequences, *Proc. DARPA Image Understanding Workshop*, San Diego, January 27-29, 1992, 479-477.
- [25] S. Sull and N. Ahuja, Integration of Multiple Features for 3D recovery and Visualization of Flight Image Sequences, submitted.
- [26] S. Sull, Integrated 3D Analysis and Analysis-Guided Synthesis, Ph.D. Dissertation, University of Illinois, Urbana-Champaign, 1993.
- [27] X. Hu and N. Ahuja, Sufficient Conditions for Double or Unique Solution of Motion and Structure, *Computer Vision, Graphics and Image Processing: Image Understanding*, July 1993, to appear.
- [28] M. Tuceryan, A. K. Jain and N. Ahuja, Supervised Classification of Early Perceptual Structure in Dot Patterns, *ICPR*, The Hague, Aug-Sep 1992.
- [29] C. Debrunner and N. Ahuja, Motion and Structure Factorization and Segmentation of Long Multiple Motion Image Sequences, *2nd ECCV*, May 18-23, 1992, 217-221.
- [30] E. Altman, Dynamical Systems Approach to Low Level Integration, *Proc. SPIE*, Boston, Nov 11-15, 1991.
- [31] N. Ahuja and T. Huang, IU at UI: An Overview and an Example on Shape from Texture, *Proc. DARPA Image Understanding Workshop*, Boston, April 6-8, 1988, 222-253.
- [32] N. Ahuja, IU at UI: An Overview of Research during 1988-90, *Proc. DARPA Image Understanding Workshop*, Pittsburgh, Sep 11-13, 1990, 134-140.
- [33] N. Ahuja and T. S. Huang, iU at UI: An Overview of Research During 1990-91, *Proc. DARPA Image Understanding Workshop*, San Diego, Jan. 27-29, 1992, 127-135.
- [34] J. Weng, T. S. Huang and N. Ahuja, *Motion and Structure from Image Sequences*, Springer-Verlag, 1992.
- [35] J. Weng, N. Ahuja and T. S. Huang, Matching Two Perspective Views, *IEEE Trans. PAMI*, August 1992, 806-825.
- [36] J. Weng, T. S. Huang and N. Ahuja, Motion and Structure from Line Correspondences: Closed-form Solution, Uniqueness and Optimization, *IEEE Trans. PAMI*, March 1992, 318-336.
- [37] J. Weng, N. Ahuja and T. S. Huang, Cresceptron: a Self-Organizing Neural Network Which Grows Adaptively, *Proc. International Joint Conference on Neural Networks*, Baltimore, MD, June 1992, Vol. I, 576-581.
- [38] J. Weng, N. Ahuja and T. S. Huang, Learning Recognition and Segmentation of 3-D Objects from 2-D Images, *Proc. 4th ICCV*, Berlin, Germany, May 1993, to appear.
- [39] J. Weng, N. Ahuja and T. S. Huang, Motion and Structure from Point Correspondences With Error Estimation: Planar Surfaces, *IEEE Trans. ASSSP*, vol. 39, No. 12, Dec. 1991, 2691-2717.
- [40] M.S. Lew, K.W. Wong, and T. S. Huang, Applying concept learning to matching, *11th ICPR*, 1992, August 30-Sept. 3, 1992, The Hague. The Netherlands, pp. A620-623.
- [41] A.C. She, K.D. Hjelmsted, and T. S. Huang, Stereo camera measurements of displacements in a frame structure, *11th ICPR*, 1992, pp. A708-711.
- [42] Yuncai Liu and Thomas S. Huang, Vehicle-Type Motion Estimation from Multi-Frame Images: A New Model for Vehicle Motion, *IEEE Trans. PAMI*, April, 1991, to appear.
- [43] Yuncai Liu and Thomas S. Huang, Estimating 3D Vehicle Motion in an Outdoor Scene Using Line Correspondences, *International Conf. Manufacturing Automation*, Hong Kong, August, 1992.
- [44] Yuncai Liu and Thomas S. Huang, Error Analysis for Linear Motion Estimation Using Straight Line Correspondences, *International Conf. Manufacturing Automation*, Hong Kong, August, 1992.
- [45] Yuncai Liu and Thomas S. Huang, Comparison of Different Methods of 3D Motion Estimation Using real Scene Image Sequence, *2nd Singapore International Conference on Image Processing*, IEEE Singapore Section, Singapore, 7-11 Sep. 1992.
- [46] M. K. Leung and T. S. Huang, Detecting Wheels in Vehicles, *Pattern Recognition*, 1992, Vol. 24, No. 12, pp. 1139-1151, 1991.
- [47] T. S. Huang, S. Reddy and K. Aizawa, Modeling Analysis and Visualization of Human Facial Motion, *Proc. SPIE Conf. on Visual Communication and Image Processing*, Nov. 11-15, 1991, Boston, MA, Vol. 1605, pp. 234-241.
- [48] Jialin Zhong, Juyang Weng, Thomas S. Huang, Vector field interpolation using robust statistics, *Proc SPIE Conf. on Curves and Surfaces in Computer Vision and Graphics II*, Boston, Nov. 11-15, 1991.
- [49] Jialin Zhong, Juyang Weng, Thomas S. Huang, Robust and Physically-Constrained Interpolation of Fluid Flow Fields, *IEEE Conf. on ASSP*, San Francisco, March 22-26, 1992, pp. III 273-276.
- [50] C.W. Chen and T.S. Huang, Analysis of left ventricle global deformation based on dynamic CT data, *International Conf. Pattern Recognition*, August 1992, The Hague, The Netherlands, pp. C443-446.

- [51] C.W. Chen, T.S. Huang and M. Arrott, Modeling Analysis and Visualization of Left Ventricle Shape and Motion by hierarchical decomposition, *IEEE Trans. Pattern Anal. Machine Intell.*, to appear.
- [52] C.W. Chen and T.S. Huang, Left ventricle motion analysis by hierarchical decomposition, *Proc. ASSP'92*, March 1992, San Francisco, CA, pp. III 185-188.
- [53] C.W. Chen and T.S. Huang, Surface modeling in heart motion analysis, *Proc. SPIE Conf. on Curves and Surfaces in Computer Vision and Graphics II*, November 1991, Boston, MA.
- [54] R. J. Qian and T. S. Huang, Motion Analysis of Articulated Objects, *ICPR*, August 1992, The Hague, The Netherlands, pp. A220-223.
- [55] J. Weng and T. S. Huang, Complete Motion and Structure from Two Monocular Sequences without Stereo Correspondence, *Proc. 11th ICPR*, The Hague, The Netherlands, Aug. 1992, pp. 651-654.
- [56] Nguyen, T. C. and Huang, T. S., Quantization Errors in Axial Motion Stereo on rectangular tessellated image sensors, *Proc. 11th ICPR*, The Hague, August 31-September 3, 1992.
- [57] Nguyen, T. C. and Huang, T. S., Image blurring effects due to depth discontinuities: blurring that creates emergent image details, *Proc. 2nd ECCV*, Italy, May 1992, 347-362.
- [58] Nguyen, T. C. and Huang, T. S., Image blurring effects due to depth discontinuities: blurring that creates emergent image details, *Image and Vision Computing Journal*, Vol. 10, No. 10, 689-698.
- [59] J. Zhong and T.S. Huang, Interpolation of Multidimensional Random Processes with Application to turbulent Fluid Flows, *Int. Conf. on Acoustics, Speech and Signal Processing*, 1993, to appear.
- [60] L. Tang, M. Pouyat, K. Aizawa and T. S. Huang, Accuracy of Modeling a Person's Face Using a Generic Model, *Proc. Picture Coding Symposium*, 1993, to appear.
- [61] M. S. Lew, T. S. Huang and K. W. Wong, Learning and Feature Selection in Stereo Matching, *these proceedings*, 1993.

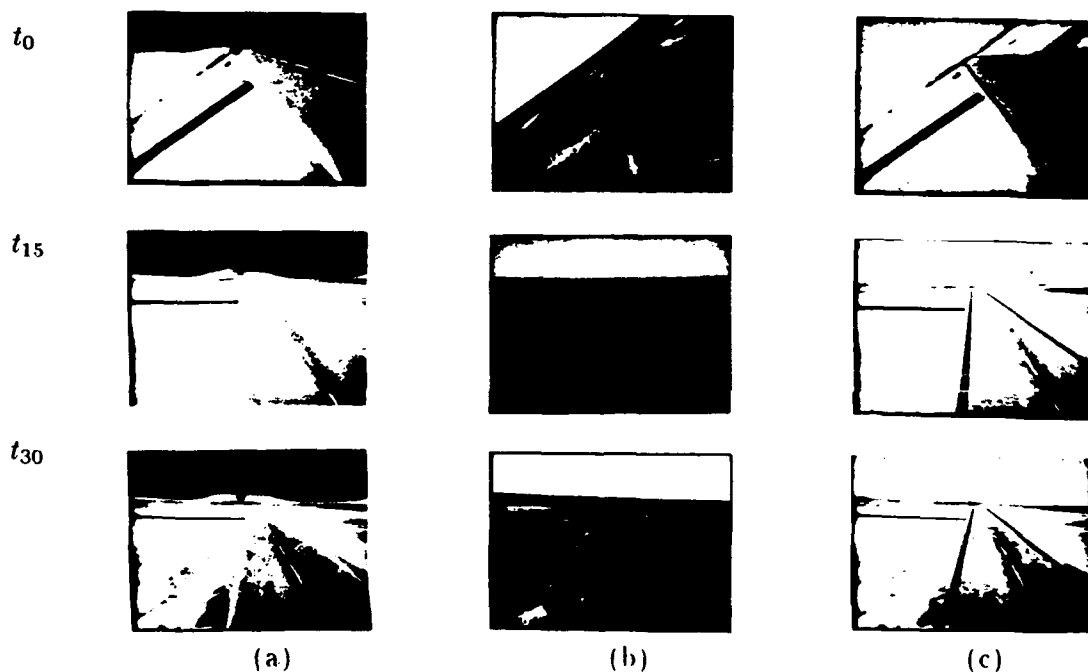


Figure 1: Analysis guided synthesis of a take-off sequence taken from a commercial laserdisc. Results are shown for three frames. (a) Input sequence. (b) Synthesized sequence composed of those image attributes selected during integrated 3D analysis. (c) Same as input sequence but the recognized runway edges are enhanced by repainting them and placing yellow disks in them. As results of the integrated 3D analysis: the airplane could be removed from the image sequence, the occluded scene parts filled in (by interpolating the estimated nearby structure), and the scene part above the vanishing line colored (blue).

Vision-Based Navigation*

William B. Thompson
Department of Computer Science
University of Utah
Salt Lake City, UT 84112

Herbert L. Pick, Jr.
Center for Research in Learning
Perception, and Cognition
University of Minnesota
Minneapolis, MN 55455

Abstract

Navigation in outdoor terrain is difficult due to a lack of easily and uniquely identifiable landmarks. This paper outlines current research on extraction of navigationally salient features from images and maps, feature matching and viewpoint determination, landmark selection, detection and diagnosis of route following errors, perceptual issues related to vision-based navigation, and database and software availability.

1 Introduction.

Navigation involves two closely related tasks: *localization* and *route planning and following*. Most often, a navigating agent has available a map or some other model of the environment within which it is operating, together with sensor data about relevant aspects of that environment at the current instant in time. Localization finds the agent's position within the map or model frame of reference. Route planning involves the determination of a sequence of actions aimed at accomplishing some goal. This may be based in part on sensor data or completely on the map or model if they are sufficiently rich. Route following includes those processes which execute the plan and monitor for errors. These activities must be closely integrated. For example, accurate localization estimates are needed for route planning since an initial position is usually required and for route following to provide closed loop control of position.

Image understanding approaches to localization must necessarily contain three parts: *feature extraction*, *matching*, and *viewpoint inference*. Feature extraction involves the detection of salient patterns in both sensed data and the map or model. Extracted features are then matched, establishing a correspondence between the two frames of reference. Finally, this correspondence is used to place the viewpoint in the map/model frame of reference. At least in principle, these steps are relatively straightforward when downward looking aerial imagery is matched against a standard "plan view" map. (TERCOM is a classic example [Andreas *et al.*, 1978]). Features

can be either raw data or simple, derived point or contour properties. Matching is essentially 2-D correlation. Viewpoint determination involves standard methods from photogrammetry.

Localization is much more difficult when performed at or near ground level due to the 90° change in perspective from sensed data to map. Passive image understanding techniques are likely to have serious problems estimating range to environmental features and thus the relative position of those features to each other and to the viewpoint in the map frame of reference. More sophisticated feature extraction and matching is required and viewpoint determination methods must be able to function in the absence of accurate 3-D information from sensors. We have made progress in the following areas:

- *Feature extraction*: Domain specific feature extraction routines have been demonstrated which exploit constraints imposed by the geometry of terrain.
- *Matching and viewpoint determination*: Higher-level symbolic problem solving has been integrated with lower-level computer vision methods to produce an image understanding system capable of dealing with inference and ambiguity in localization.
- *Landmark selection*: Path following is significantly aided by selecting landmarks which minimize localization errors.
- *Diagnosis and recovery*: AI-like problem solving methods can complement lower-level computer vision in detecting failures in route following and diagnosis where the original error occurred.
- *Perceptual issues*: An understanding of the abilities and limitations of human perception of terrain features can give insights into the construction of automated navigation and also lead to better training methods.
- *Database*: Examples of panorama images registered to digital elevation data together with a variety of useful software tools are being made available to the research community.

Results of this work are of potential relevance to autonomous and semi-autonomous mobile vehicles, navigation aids, mission planning, simulation, and training.

*This work was supported by National Science Foundation grant IRI-9196146, with partial funding from the Defense Advanced Research Projects Agency.

2 Localization.

Our work on navigation has focused primarily on problems involving outdoor, unstructured terrain. Figure 1 shows typical feature correspondences that must be established. Since distinctive cultural landmarks are not available in such environments, considerable difficulties can be expected in reliably associating map and view features. One way to approach this problem is to use symbolic matching by first independently extracting from the view and map patterns likely to represent the same topographic features and then establishing correspondences using a hypothesize and test strategy.

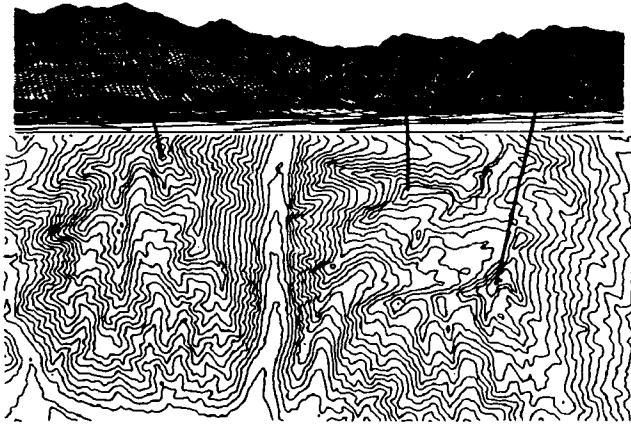


Figure 1: Correspondence between map and view.

Feature extraction from images of outdoor terrain is based on finding ridge contours with shapes indicative of peaks, saddles, and valleys. Peaks and saddles are simply vertical extrema in ridge line contours. Valleys are more difficult to find, since the actual valley terrain is usually not visible and must be inferred from other features such as T-junctions in ridge line contours.

Simple edge detection alone is not sufficient to find ridge contours in an image. Images of large-scale, outdoor terrain contain many important but indistinct features and many extraneous features which convey no useful information about the topography. The contrast across ridge contours is often low and of limited spatial extent. Often, local sections of a ridge contour are lacking in contrast variation altogether, while many non-ridge, high-contrast features are present.

Figure 2 shows a 40° portion of the panorama image shown in Figure 11. Figure 3 shows the results of applying a zero-crossing edge detector to this image. Hysteresis thresholding was used and parameters were carefully matched to the nature and scale of the image. As a result, this represents about the best that can be expected from edge detection alone. Figure 4 shows a new edge image in which a variety of filtering and gap filling steps have been applied. These steps are based on exploiting constraints about how ridge lines appear in horizontally-looking views of rugged terrain. Finally, Figure 5 shows extracted features and line segments.

Figures 6 and 7 show similar results for map features. Unlike the problem of extracting topographic structure

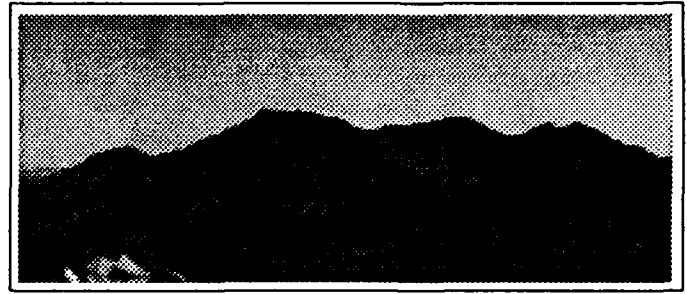


Figure 2: Original image.



Figure 3: Output from zero-crossing edge detector.



Figure 4: Processed edges.

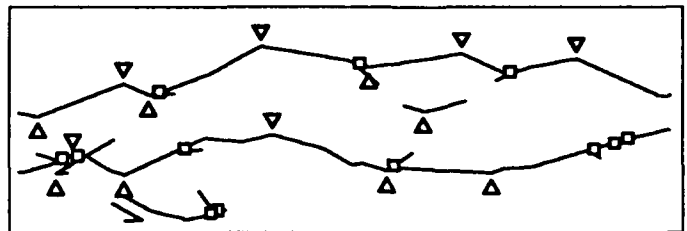


Figure 5: Extracted features.

from images, the "map-understanding" problem does not have to deal with the multitude of effects that can lead to contrast variation in images. Difficulties associated with scale are still very real, however. For example, ridge lines have a large spatial extent along their length. Across the length of a single ridge line, extent can vary from small (a sharp section of ridge) to quite large (a section where the ridge top is essentially a plateau). Peaks are likewise more difficult to accurately detect. Simply finding local maxima in elevation is not sufficient. Figure 6 shows the results of applying a local ridge detector similar to [Haralick *et al.*, 1983] to a portion of our elevation database (see section 6). Figure 7 shows the final results of feature extraction after thinning the raw results and filling gaps where ridge sharpness was low. In addition, peaks

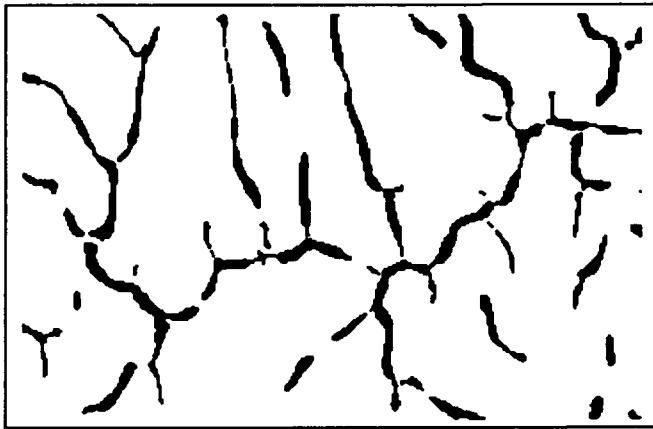


Figure 6: Unprocessed ridge features.

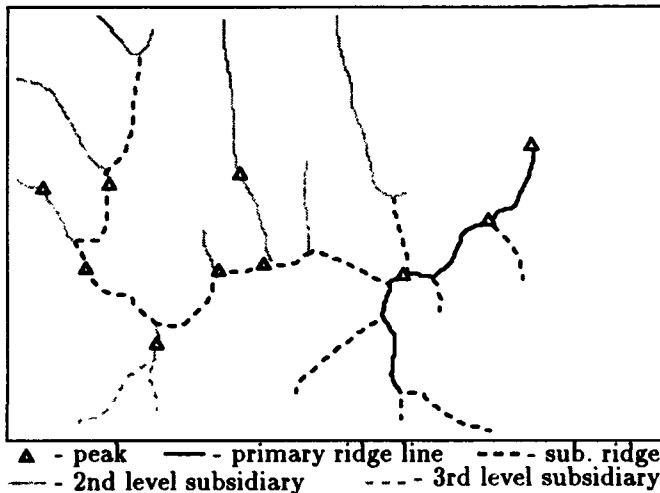


Figure 7: Peaks and ridge line hierarchy.

are found using a large area search that is more reliable than simple local maxima detection and ridge lines are organized into a hierarchy of importance that allows significant ridges to be used for initial matching while making available subsidiary ridges for subsequent verification operations. (The ridge lines to the northwest are not rendered in this view of the hierarchy, since they are actually part of the parent of the ridges shown.)

Features extracted using these processes still have a great deal of ambiguity associated with them. For example, lacking a priori information about viewing position and/or direction, it is hard to extract features such as peaks and ridges known to correspond in the view and map. This difficulty is similar to that faced by many symbolic problem solving systems dealing with tasks such as classification and diagnosis. In [Thompson *et al.*, 1993], we show that high-level hypothesize and test strategies can be integrated with lower-level feature extraction to solve difficult localization problems.

Additional details about feature extraction from views and maps can be found in [Savitt *et al.*, 1992, Savitt, 1992, Thompson *et al.*, 1993]. High-level strategies for feature matching are described in [Heinrichs *et al.*, 1989,

Thompson *et al.*, 1990, Smith *et al.*, 1991, Heinrichs *et al.*, 1992] and computational implementations using these strategies can be found in [Bennett, 1992, Bennett, 1993, Thompson *et al.*, 1993, Thompson, 1993].

3 Landmark Selection.

We have previously demonstrated that the accuracy of landmark-based viewpoint determination is quite sensitive to geometric properties of the particular configuration of landmarks used [Sutherland, 1992]. Recently, the image understanding community has been paying increased attention to error estimation. Of equal importance are approaches which minimize the amount of error which can occur rather than only providing a posteriori characterizations of the error distribution.

The extraction of navigationally salient landmarks typically involves costs in time, computation, and sensing resources. As a result, there is benefit to be gained if simple strategies can be used to select a small set of landmarks which are likely to lead to accurate localization. Effective landmark selection methods are also relevant to mission planning, where one of the criteria entering into route selection should be the availability of landmarks sufficient to provide whatever degree of accuracy is required.

Error analysis is complicated by the lack of general sensor models which effectively describe position variability in properties used for viewpoint determination. This is particularly true when localization is based on bearings to features over a wide field of view, since sensing might involve mechanical scanning of cameras, fish eye optics, or more exotic technologies. We take a conservative approach in which we assume that the angular error in detected bearings to features is bounded, but the distribution of values within this range is not known. We then find the region within which the viewpoint must lie to be consistent with these assumptions and are thus able to determine if conflicts with obstacles or untraversable terrain are possible. Figure 8 shows an example in which the relative bearing between two landmarks and the absolute bearing to a third landmark [Thompson *et al.*, 1993] separately generate possible viewpoint regions shown in light gray, the intersections of which are marked in dark gray.

Starting from the analysis of uncertainty regions, it is possible to develop simple heuristics for selecting landmarks likely to minimize the size of such regions [Sutherland and Thompson, 1993, Sutherland, 1993]. Note that

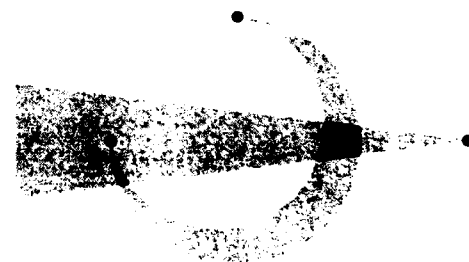


Figure 8: Intersection of viewpoint uncertainty regions.

this is not as easy as it might seem, since the problem must be solved with very minimal knowledge about the true viewing location. Figures 9 and 10 demonstrate the effectiveness of this method. Simulated navigators have identified landmarks on a map. Their task is to move along the segmented path shown by the dashed line. Current position is estimated at the beginning of each straight path segment, using relative bearing to three landmarks. In Figure 9, the landmark selection heuristic is used at each step to choose the three landmarks on which localization is based. In Figure 10, landmark selection is random. Both navigators start at the square at the left end of the dashed line. Direction and distance of move are based on estimated position. Uniform multiplicative error is assumed in both relative bearing measurement and in movement. The squares mark actual navigator positions at the end of each path segment for fifty trials. The scattering of location in Figure 10 is much increased over that in Figure 9.

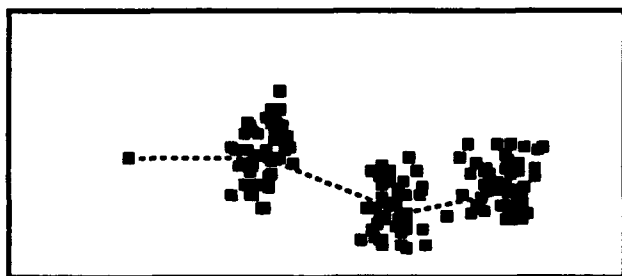


Figure 9: Fifty trials - "intelligent" landmark selection.

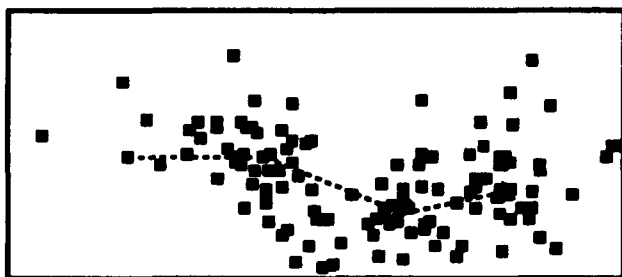


Figure 10: Fifty trials - random landmark selection.

4 Error Detection and Diagnosis.

Mobile robots capable of independent operation all employ some form of "perceptual servoing" to implement a sense-plan-act-verify cycle in which expectation about sensor data are compared with actual observations, and then differences are quantified and used to update estimates of current position and desired path (e.g., [Fennema *et al.*, 1990]). If a match between expectations and observations cannot be established, then some sort of re-planning activity is initiated, a part of which requires a solution to the localization problem. This approach is only effective when a rich model of the environment is available, allowing for complete and specific predictions about the appearance of the world from any predicted

viewpoint. Often, such models do not exist, particularly in tasks involving outdoor maneuvering. (Consider the effort that went into producing 5m resolution DEM data for the ALV site.) When this is the case, it is not possible to determine with certainty that an expectation does or does not match actual sensor values. At best some sort of confidence estimate can be produced. One consequence of this is that it is possible to travel substantial distances on what is in fact an incorrect path before determining with reasonable certainty that an error has occurred.

Sparse world models and the potential for substantial delays between when an error occurs and when it is detected mean that lower-level image understanding techniques are not sufficient in and of themselves to support effective plan monitoring in mobile robotics. We are addressing this problem by creating a qualitative model of error in vision-based navigation and using this model to characterize the sorts of errors that can occur, how they can be detected, and what sort of diagnosis is possible to determine the original source of difficulties [Stuck, 1992]. The research suggests a number of techniques that may usefully complement lower-level perceptual servoing.

5 Perceptual Issues.

Our approach to the development of novel methods for vision-based navigation is interdisciplinary, involving computational analysis, computer simulations, and studies of expert map users. Many of the strategies we use to automatically solve localization problems [Heinrichs *et al.*, 1992, Thompson *et al.*, 1993] arose out of experiments done with experts solving actual and artificial navigation problems [Pick *et al.*, in press]. In retrospect, these strategies make excellent computational sense since the experts are highly adapted to dealing with the ambiguity and complexity inherent in these problems. Nevertheless, the strategies were not obvious to us or others until we undertook our studies.

This interdisciplinary investigation is continuing with a current focus on the accuracy with which people are able to determine terrain geometry. By comparing human and machine vision perceptual competence, we can better understand the relevance of expert strategies to image understanding solutions. At the same time, we can identify specific perceptual skills for which mechanized aids and/or alternative training might significantly improve human performance. Elsewhere in this proceedings we summarize two such studies [Pick *et al.*, 1993]. One demonstrates that people are poor at estimating distance and slope in environments of the scale and topography typical of outdoor navigation tasks [Melendez *et al.*, in prep]. Since passive vision systems are also poor at these estimates, the strategies people use to compensate for their perceptual limitations may also be relevant in automated systems. The second study deals with localization using visual angle. Again, people are quite poor at using this cue. On the other hand, sensors which are capable of measuring large visual angles with reasonable accuracy can be designed, suggesting both possible differences between machine and human solutions and aids that might assist people in performing the task.

Figure 11: First panorama image.

6 Database and Software.

Many recent papers addressing ground level localization have presented results obtained only from synthetic imagery. A few have used the highly calibrated data available for the Martin Marietta ALV test area. In addition to the well-known pitfalls of failing to test new image understanding algorithms on real data, the use of synthetic terrain data to generate test imagery is problematic since realistic digital elevation data is often in error.

We have produced two 360° panorama images of mountainous terrain obtained with a video camera, digitized at high resolution, and digitally photo-mosaicked. (Figure 11 shows one of them). They extend approximately 6,000 pixels horizontally by 450 pixels vertically. Viewpoint location has been registered ± 30 m to USGS 30m DEM data. Direction relative to UTM north and tilt are known within $\pm 0.5^\circ$. Geometric distortions due to misalignments between the pan axis, the camera, and "true" vertical have been normalized to approximately $\pm 0.25^\circ$. Included in the database are 4 USGS 7.5' DEM quadrangles composed together and containing the viewpoints for the panorama images. Also available is software for converting USGS format data into a useful form, mosaicking DEM quads and panorama frames, and rendering expected views given map position.

References

- [Andreas *et al.*, 1978] R.D. Andreas, L.D. Hostetler, and R.C. Beckmann. Continuous Kalman updating of an inertial navigation system using terrain measurements. In *Proc. IEEE National Aerospace Electronics Conference*, pages 1263–1270, 1978.
- [Bennett, 1992] B.H. Bennett. *A Problem-Solving Approach to the Localization Problem*. PhD thesis, University of Minnesota, 1992.
- [Bennett, 1993] B.H. Bennett. Knowledge-based control for robot self-localization. In *Proceedings NASA Goddard Conference on AI applications to Space*, May 1993.
- [Fennema *et al.*, 1990] C. Fennema, A. Hanson, E. Riesenman, J.R. Beveridge, and R. Kumar. Model-directed mobile robot navigation. *IEEE Trans. on Systems, Man and Cybernetics*, 20:1352–1369, November/December 1990.
- [Haralick *et al.*, 1983] R.M. Haralick, L.T. Watson, and T.J. Laffey. The topographic primal sketch. *IEEE Journal of Robotics and Automation*, 2(1):50–72, 1983.
- [Heinrichs *et al.*, 1989] M.R. Heinrichs, D.R. Montello, C.M. Nusslé, and K. Smith. Localization with topographic maps. In *Proceedings of the AAAI Symposium on Robot Navigation*, pages 29–32, March 1989.
- [Heinrichs *et al.*, 1992] M. R. Heinrichs, K. Smith, H. L. Pick, Jr., B. H. Bennett, and W.B. Thompson. Strategies for localization. In *Proc. DARPA Image Understanding Workshop*, January 1992.
- [Melendez *et al.*, in prep] P.H. Melendez, D.A. Gentile, Jr. Pick, H.L., A. Yonas, and D.J. Wegesin. Psychophysical judgments of natural terrain. (in prep.).
- [Pick *et al.*, 1993] H.L. Pick, Jr., A. Yonas, P. Melendez, D. Wagner, D. Gentile, and D. Wegesin. Perceptual aspects of navigation. In this proceedings, 1993.
- [Pick *et al.*, in press] H.L. Pick, Jr., M.R. Heinrichs, D.R. Montello, K. Smith, C.N. Sullivan, and W.B. Thompson. Topographic map reading. In J. Flach, P.A. Hancock, J.K. Caird, and K. Vicente, editors, *Ecology of Human-Machine Systems*. Lawrence Erlbaum Associates, (in press).
- [Savitt *et al.*, 1992] S.L. Savitt, T.C. Henderson, and T.L. Colvin. Feature extraction for localization. In *Proc. DARPA Image Understanding Workshop*, 1992.
- [Savitt, 1992] S.L. Savitt. *A Context Sensitive Segmentation Approach for Outdoor Terrain Feature Extraction*. PhD thesis, University of Minnesota, 1992.
- [Smith *et al.*, 1991] K. Smith, M.R. Heinrichs, and H.L. Pick, Jr. Similarity judgment and expert localization. In *Proceedings Thirteenth Annual Conference of the Cognitive Science Society*, August 1991.
- [Stuck, 1992] E.R. Stuck. *Detecting and Diagnosing Mistakes in Inexact Vision-based Navigation*. PhD thesis, University of Minnesota, 1992.
- [Sutherland and Thompson, 1993] K.T. Sutherland and W.B. Thompson. Inexact navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 1993.
- [Sutherland, 1992] K.T. Sutherland. Sensitivity of feature configuration in viewpoint determination. In *Proc. DARPA Image Understanding Workshop*, pages 315–319, January 1992.
- [Sutherland, 1993] K.T. Sutherland. Landmark selection for accurate navigation. In this proceedings, 1993.
- [Thompson *et al.*, 1990] W.B. Thompson, H.L. Pick, Jr., B.H. Bennett, M.R. Heinrichs, S.L. Savitt, and K. Smith. Map-based localization: The "drop-off" problem. In *Proc. DARPA Image Understanding Workshop*, pages 706–719, September 1990.
- [Thompson *et al.*, 1993] W.B. Thompson, T.C. Henderson, T.L. Colvin, L.B. Dick, and C.M. Valiquette. Vision-based localization. In this proceedings, 1993.
- [Thompson, 1993] W.B. Thompson. Geometric constraints for viewpoint determination. University of Utah Computer Vision Group working paper, 1993.

Progress in Image Understanding Research at Brown University *

David B. Cooper [†], Thomas L. Dean [‡], William A. Wolovich [†]

[†] Division of Engineering, [‡] Department of Computer Science,
Brown University, Providence, RI 02912, USA

email: cooper@lems.brown.edu, tld@cs.brown.edu, waw@lems.brown.edu

Abstract

Work progressed on many fronts this year. A primary focus has been on the assembly of an indoor service robot which has the capability of searching for a specified free-form object in a cluttered environment. The robot mounts a number of sensors — stereo and laser ranging — and has model-based planning capability to optimize use of its resources. This system, which is now functioning, is undergoing testing and continuing development. This project is a collaborative effort by the P.I.s. In addition, vigorous research programs have been continuing in computer vision, planning, and mechanical sensing. This report summarizes our program for the year. Quite a few papers were published during the year. A number of them are referenced in this paper. Others are referenced in our technical papers in this workshop proceedings.

1. **Vision:** A powerful new technology for recognizing free-form 2D and 3D objects has now been brought to a usable state. This involves fitting a high degree implicit polynomial to the data, computing a vector of invariants of the polynomial coefficients, followed by a minimum probability of error comparison of that vector with a vector of invariants stored in a data base for each possible object to be recognized. A new geometric-stochastic approach has been developed for completely automated estimation of main roads and similar structures such as rivers in aerial images. An approach has been developed for the joint estimation of 3D structure and camera motion based on finding corresponding regions in two images through use of new affine moment invariants and solving simple motion equations explicitly.

Considerable progress has been made on a deterministic approach to shape by modeling deformations of it by Hamilton-Jacobi and reaction-diffusion equations, and has led to a novel theory of partitioning of visual form and a geometric evolutionary view of mathematical morphology, among others.

2. **Planning and System Integration:** A mobile robot carrying infrared proximity sensors, a laser light-stripe range sensor, and a pair of stereo cameras is now functional. The system can recognize free-form objects and make optimal use of the robot's sensing routines to search a cluttered environment for objects of a specified type.

3. **Object Recognition via 3-D Surface Tracking:** A 3-D dual-drive surface tracking controller that enables a robot to track along any specified path on the surface of an unknown object in order to identify the object is under development and testing. The dual-drive controller computes the normal and tangent vectors relative to movement along the path. The result is controlled movement in 3-D on the surface of an object. It

is assumed that the path is generated by an external recognizer in such a way that the data points collected by tactile sensing along the path will maximize the probability of correctly identifying the object.

1 Vision

1.1 Free-Form 2D and 3D Object Recognition Based on Implicit Polynomials, Algebraic Invariants, and Asymptotic Bayesian Methods.

We have now brought our free-form object recognition based on implicit polynomials and algebraic invariants to a useful stage where we can begin to apply it to real problems. At present, the system can deal with the recognition of which of L known objects is present in 3D range data when the object in the range data is in arbitrary position, i.e., location and orientation. Examples of this are object recognition based on LADAR range data or 3D data from stereo. Exactly the same technology handles the recognition of which of L object boundaries is present in image edge data when the camera view direction is arbitrary so that the boundary in the image is in arbitrary position and will usually have undergone different scale changes in each of two directions. Examples of this problem are ground target recognition based on a silhouette in an aerial image, or recognition of an airborne plane based on a silhouette in an image by a ground-based camera. Our recognizer fits one general implicit polynomial to the data, computes a vector of invariants for the polynomial (these are functions of the polynomial coefficients that are functions of shape only and are invariant to object position and stretchings in two different directions), and does a Bayesian comparison of this vector with a stored vector of invariants for each object in the database.

This recognizer has two crucially important features! First, it functions well even if data is over only a portion of an object boundary due to self occlusion or occlusion by another object. Second, the required computation is small—linearly proportional to the number of data points used in the recognition. Details of the approach and a number of examples are given in [22] in this proceedings.

One application of this approach is the following. An approach to target recognition is to decompose a target into parts invariantly, i.e., a decomposition that is invariant to partial occlusion or to changes in viewing direction. Such a decomposition is that of

*This work was partially supported by NSF-DARPA Grant #IRI-8905436

the silhouette of an F-16 using an approach developed by Kimia et. al. [19]. The parts obtained by the hierarchical decomposition are shown in Figure 1(a). The final partitioning of the plane is shown in the upper right corner of the image. These parts would then be recognized by our recognizer, and the results combined for target recognition. All of the final parts can be fit *with negligible error* by 4th degree implicit polynomial curves, and it is practical to go to higher degree if needed. Fits to a representative group of these airplane parts are displayed in Figure 1(b), where it is almost impossible to distinguish the part boundaries and the fitted curves. If a more complex part is to be represented, e.g., a rocket at a wingtip, a 4th degree polynomial may provide a fit of only modest accuracy, as shown. In such a case, a higher degree polynomial can be used, e.g., as shown, an 8th degree polynomial fits the data with negligible error. The computational cost of fitting an 8th degree polynomial is roughly that of fitting a 4th degree polynomial. We are also exploring the recognition of very complex objects using patches of low degree polynomials without the need of repeatable segmentation.

1.2 Completely Automated Recognition and Estimation of Main Roads in Aerial Images.

Our ultimate goal is recognition and estimation of all structure of interest in aerial imagery. However, to start we are focussing on main roads and similar structure such as rivers, etc., because we can build on technology which we have developed previously. Most algorithms in the published literature require an operator to give a pair of initial road boundary points. This interaction simplifies the problem of road estimation tremendously. We are after a completely autonomous system. We build on our earlier work on blob boundary finding that included the introduction of "stochastic snakes", which we termed "the ripple filter", the Cramer-Rao lower bound on the minimum achievable error variance in estimating blob boundary location, and a dynamic programming algorithm for implementing maximum a posteriori probability blob boundary estimation [6, 8]. We use stochastic-geometric models and model road geometry and image intensity inside and outside the roads by autoregressive processes which are well suited to both image synthesis and road estimation. We find all of the main roads in an image, irrespective of their intersections, variability in width, lack of image intensity discontinuity at some groups of boundary points, whether or not they have visible barriers, etc.. This work is discussed in [2] in this proceedings.

1.3 Estimation Of 3D Surfaces And Camera Motion From Two Or More Images.

3D surfaces are to be estimated from two images. It is assumed that the position of the camera at which each image is taken is completely arbitrary and a priori unknown. This occurs if a camera is moving in an unknown way or if images are taken by two or more cameras distributed through a 3D region and moving locally in an unknown way. Our approach is to approximate an arbitrary 3D surface with small planar

patches where each such patch has location and orientation that is to be estimated, and simultaneously estimate camera position 2 with respect to camera position 1. We do this by solving a set of explicit equations for the unknown parameters in terms of low computational cost, stable measurements that we make on the images. These measurements require matching — finding a corresponding region in image 2 for each region in image 1. Since the image in such a region in image 2 may be a distortion of that in the corresponding region in image 1 because of the differences in camera viewing directions, we use affine moment invariants for carrying out the matching. *This approach is computationally attractive and can be used more generally for aligning two or more aerial images.* The approach seems to work well, and is described in [17] in this proceedings and in [18].

1.4 A Hamilton-Jacobi Approach to Recognition

We have made substantial progress in our "shape from deformation" framework [14, 12] which is based on shocks formed from reaction-diffusion and Hamilton-Jacobi equations. We have shown that algebraic, set-theoretic mathematical morphology operations with any convex structuring element can be viewed and implemented as geometric evolution equations governed by a Hamilton-Jacobi partial differential equations [1]. We had earlier shown that Gaussian smoothing is a special case of this framework as well. The general approach has also motivated successful application to shape-from-shading [15]. Robust recognition of shape requires a multi-dimensional representation of it in terms of its parts, protrusions, and bends. In the past year, we have developed a theory of partitioning for visual form, which is based on general assumptions about object formation and projection. Our notion of parts is based on notions of *necks* and *limbs* and is supported by computation, psychophysical and ecological constraints. The decompositions give natural intuitive parts that are in correspondence with functional three-dimensional parts for a range of biological and man-made shapes [23, 20]. We have successfully applied this scheme to military targets in LADAR imagery. We are currently studying protrusions and bends, the other two nodes of the *shape triangle* [13], necessary to describe shape for robust recognition.

2 Mobile Robot Project

2.1 Mobile Robot Planning

Over the past year, we have made a concerted effort to combine our work in image understanding, planning, and control. To that end we have designed and constructed a new mobile robot, developed software to control the robot and interpret the data returned by its sensors, and begun conducting experiments to evaluate our hardware and software. This work builds on our experience with a smaller robot, extending the same basic architecture [9] and incorporating more sophisticated image understanding techniques. In the following, we briefly summarize this work; a more detailed account is available in these proceedings [4].

Our research has been driven by a specific class of tasks. These tasks involve navigation, obstacle avoidance, and object recognition and focus primarily on efficiently searching in cluttered environments for objects specified in some suitable representation language. In a typical task, the robot is confined to an area of about 1000 square feet, cluttered with obstacles and containing a number of target objects corresponding to several known object types. The robot is given a particular object type and required to find an instance of that type in the enclosed area. The robot's search is methodical but the robot does not spend time on computationally expensive information gathering operations when less expensive operations suffice.

While the emphasis is on employing the image understanding algorithms developed in our overall effort, we use a variety of sensors to expedite search. Familiar with the advantages and disadvantages of using sonar, we decided to complement our acoustic ranging capability with near-infrared obstacle detection and laser-light-stripe ranging. In addition, the robot has a fixed, forward-directed stereo pair. The sensors and computing machinery for the control system were built on a heavy-duty 24 inch mobile base that provides power for experiments running up to six hours (this represents a 3-5 fold increase in battery life with about a 10 fold increase in onboard computing power).

We integrated the sensors into a set of robust navigation routines that comprise the low-level control system. The low-level control and sensor fusion algorithms were implemented as a set of objects and methods in C++. Due to the modularity of our software design and the similarity of the design of the new base with that of our earlier base, much of our existing C++ software could be directly transferred to the new robot.

The high-level planning system is based on our work on *temporal belief networks* [10] which is designed to address a range of planning and control problems [16, 3]. We are now extending and refining our techniques to handle more complicated stochastic models describing the dynamics of the domain and the characteristics of sensors. In particular, we are working on planning applications that make use of the information returned from object-recognition routines.

The tasks that we are focusing on require the robot to search for and recover an object of a specified type in a cluttered environment. In order to search efficiently, the robot has to deploy its sensors carefully. Laser ranging takes a few milliseconds, a quick-and-dirty analysis of an image to identify possible locations of the target takes a few seconds, and a more careful matching against a prototype takes 15-30 seconds. Planning involves, among other things, choosing from among a set of information-gathering strategies so as to expedite search for the target object.

Current research involves the development of a programming environment that facilitates the design and compilation of planning systems implemented as temporal belief networks. We anticipate that by the end of the first half of 1993 we will be able to produce complete planning systems using a semi-automated, interactive system in a small fraction of the time required previously.

2.2 Mobile Robot Free-Form Object Recognition Based on Stereo Vision

A 3D object is to be recognized from a stereo-pair of images by estimating a portion of the 3D object surface and then applying the recognizer from section 1.1. Objects of interest can be anything from simple polyhedra to complicated free-form solids.

Our approach to estimating a surface from two or more images involves modeling the surface as a smooth stochastic process with occasional depth discontinuities, and estimating the surface directly from the images using maximum a posteriori probability estimation. We design and use an appropriate Markov Random Field (MRF) as a prior model for the surfaces. Though the field can be designed to capture any desired structure, in the system used in these experiments the purpose of this field is largely to act as a smoother (i.e., regularization). Details of the algorithm are given in [7, 5, 11].

Having obtained 3D surface points, object recognition is done by fitting a fourth degree implicit polynomial surface to these points, and then comparing the vector of algebraic invariants for this polynomial with stored vectors of invariants. Comparison is done using a Bayesian recognizer. *The beauty of using this recognizer is that it is roughly equivalent to checking how well the set of 3D estimated surface points matches the stored model, and hence, works excellently even if the set of points is over only a portion of the object surface due to occlusion!* Details of the Bayesian approach and other references are given in [22, 21].

Figure 2 shows the various steps in the algorithm. Figures 2(a) and 2(b) are images of the object. The object is located in a busy environment. Table legs, wires, a file cabinet and a person's legs can be seen in these images. The reconstructed surface produced by the stereo algorithm is shown in Figure 2(c). Object recognition is done using the Bayesian recognizer. Final verification of the recognition, if needed, is done by translating and rotating the data set so as to fit the database model as shown in Figure 2(d).

The problem with recognition based on the data from one stereo pair of images is that only a quarter to a third of the object surface is seen by both cameras, and it is therefore difficult to see enough of the curved surface to discriminate between similar shapes. Highly reliable detection is possible if 3D point estimates from two or more stereo pairs, each pair taken from a different position, can be aligned. Then estimates over more of an object surface can be used. We are presently completing software for this purpose.

3 Object Recognition via 3-D Surface Tracking

The current focus of this research has been the development and testing of a 3-D dual-drive surface tracking controller that enables a robot to track along any specified trajectory on the surface of an unknown object. In the complete "object-dependent" tracking system, we envision an external recognition program that uses partial data sets collected by tactile sensors on the object's surface to attempt an identification

and to direct future data collection in such a way as to limit the uncertainty in making a positive identification. (see, e.g., the active vision algorithm in [21]). This tactile data collection method is referred to as "object-dependent" sensing because although no prior information is used by the controller, the location of the sensing paths is driven by external sensor data or by comparisons made by a recognizer to a model data base. The application for such a data collection system is object recognition tasks in environmental exploration and manipulation.

The specification of the tracking path can be illustrated by the following. An external sensing system, i.e. vision, roughly locates bounding points for the trajectory. These points may be above, below or on the surface. The trajectory that the robot follows can be found by passing a plane along the surface normal between the bounding points. Force and velocity errors along the path are zeroed by the dual-drive controller. The result is controlled movement in 3-D on the surface of an object.

Data collection with the 3-D dual-drive controller is an improvement over the general data collection methods with the 2-D dual-drive controller. The 3-D dual-drive controller allows the robot to track along any path with the end effector in any orientation. The 2-D dual-drive controller limits tracking to mutually perpendicular horizontal and vertical planes. By using orientation coordinate transformation mappings, the 3-D tracking controller enables the robot to move in any specified tracking plane.

This controller is implemented and tested using an IBM 7565 Cartesian robot equipped with strain gauges on the end effector. The tracking controller is designed so that the robot will be able to track any free form complex object.

Figure 3(a) and 3(b) are examples of two 4th degree polynomial surfaces fit to data sensed by this system by tracking around an object along horizontal slices. One 4th degree polynomial is fit to data over an eggplant; the other is fit to data over a pear.

References

- [1] A. Arehart, L. Vincent, and B. B. Kimia. Mathematical morphology: The Hamilton-Jacobi connection. In *ICCV*, 1993.
- [2] M. Barzohar and D. B. Cooper. Completely Automatic Finding of Main Roads in Aerial Images By Using Geometric-Stochastic Models and Estimation. In *this proceedings*.
- [3] K. Basye, T. Dean, J. Kirman, and M. Lejter. A Decision-Theoretic Approach to Planning, Perception, and Control. *IEEE Expert*, 7(4):58-65, 1992.
- [4] T. Camus, J. Monsarrat, and T. Dean. Planning and Selective Perception for Mobile Robot Object Retrieval Tasks. In *this proceedings*.
- [5] B. Cernuschi-Frias, D. B. Cooper, Y. P. Hung, and P. N. Belhumer. Toward a Model-based Bayesian Theory for Estimating and Recognizing Parameterized 3D Objects Using Two or More Images Taken from Different Positions. *IEEE Trans. on PAMI*, pages 1028-1052, October 1989.
- [6] D. Cooper, H. Elliott, F. Cohen, L. Reiss, and P. Symosek. Stochastic Boundary Estimation and Object Recognition. In *Image Modeling*, Edited by Azriel Rosenfeld, pages 63-94, 1981.
- [7] D. B. Cooper, B. Cernuschi-Frias, J. Subrahmonia, and Y. P. Hung. Use of Markov Random Fields in Estimating and Recognizing Objects in 3D Space. To appear as a chapter in *Markov Random Fields: Theory and Application*, Edited by Rama Chellapa and Anil Jain, Academic Press, 1992.
- [8] D. B. Cooper and F. Sung. Multiple-Window Parallel Adaptive Boundary Finding in Computer Vision. *IEEE Trans. on PAMI*, pages 299-316, May 1983.
- [9] T. Dean and J. Kirman. Representation Issues in Bayesian Decision Theory for Planning and Active Perception. In *Proceedings of the DARPA IU Workshop*, pages 763-768, 1992.
- [10] T. Dean and M. Wellman. *Planning and Control*. Morgan Kaufmann, San Mateo, California, 1991.
- [11] Y. Hung, D. Cooper, and B. Cernuschi-Frias. Asymptotic Bayesian Surface Estimation Using an Image Sequence. *IJCV*, 6:2:105-132, 1991.
- [12] B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker. Exploring the shape manifold: The role of conservation laws. In *Proceedings of the Shape in Picture NATO conference*, September 1992.
- [13] B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker. The shape triangle: Parts, protrusions, and bends. Technical Report TR-92-15, McGill University Research Center for Intelligent Machines, 1992.
- [14] B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker. Shapes, shocks, and deformations, I: The components of shape and the reaction-diffusion space. *IJCV*, Submitted, 1992.
- [15] R. Kimmel, K. Siddiqi, B. B. Kimia, and A. Bruckstein. Shape from shading via level sets. *IJCV*, Submitted, 1992.
- [16] J. Kirman, K. Basye, and T. Dean. Sensor Abstractions for Control of Navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2812-2817, 1991.
- [17] C. Y. Lee and D. B. Cooper. Structure and Motion From Region Correspondences and Affine Invariants. In *this proceedings*.
- [18] C. Y. Lee and D. B. Cooper. Structure form motion: A region based approach using affine transformation and moment invariants. To appear in *Proceeding, IEEE Intl. Conf. on Robotics and Automation*, 1993.
- [19] K. Siddiqi and B. B. Kimia. Parts of Visual Form: Computational Aspects. Technical Report LEMS 114, LEMS, Brown University, November 1992.
- [20] K. Siddiqi and B. B. Kimia. Parts of visual form: Computational aspects. In *CVPR*, 1993.
- [21] J. Subrahmonia, D. B. Cooper, and D. Keren. Practical Reliable Bayesian Recognition of 2D and 3D Objects Using Implicit Polynomials and Algebraic Invariants. Technical Report LEMS-107, Brown University, May 1992. Under review for publication in the *IEEE Trans. on PAMI*.
- [22] J. Subrahmonia, D. Keren, and D. B. Cooper. An Integrated Object Recognition System Based on High Degree Implicit Polynomials, Algebraic Invariants, and Bayesian Methods. In *this proceedings*.
- [23] K. J. Tresness, K. Siddiqi, and B. B. Kimia. Parts of visual form: Ecological and psychophysical aspects. Technical Report LEMS 104, LEMS, Brown University, May 1992.

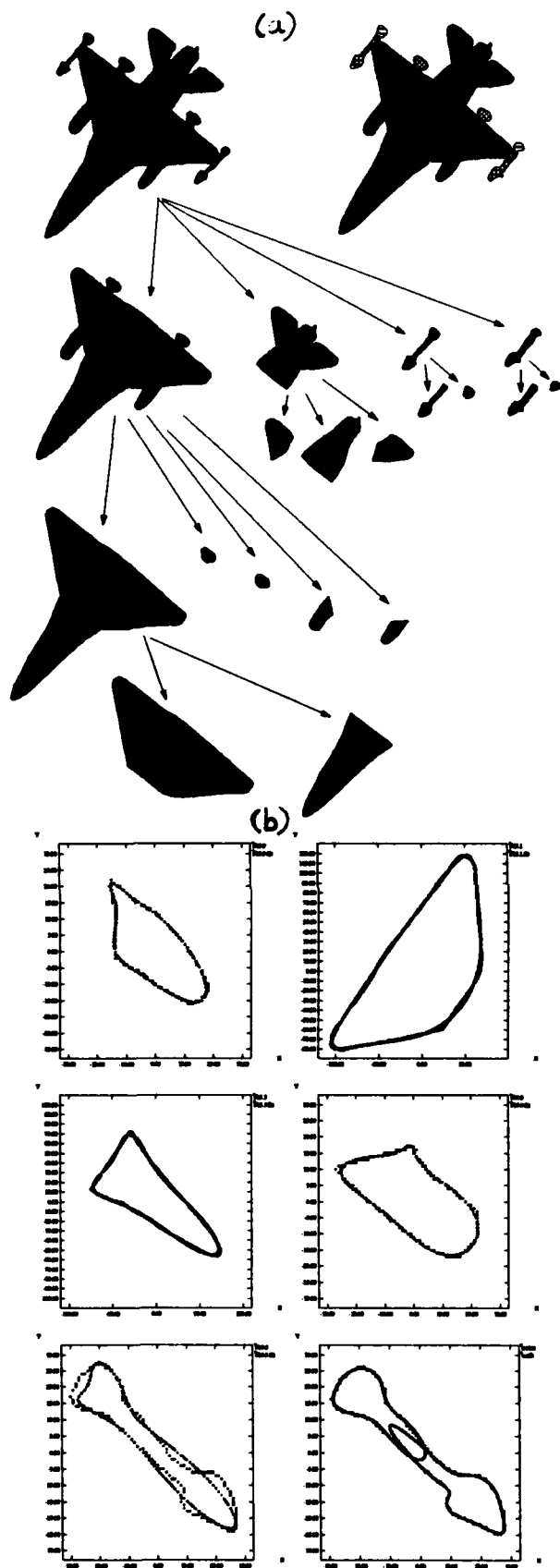


Figure 1: (a) : Final partitioning of the plane and the parts obtained by hierarchical decomposition; (b) : Polynomial fits to a representative group of airplane parts

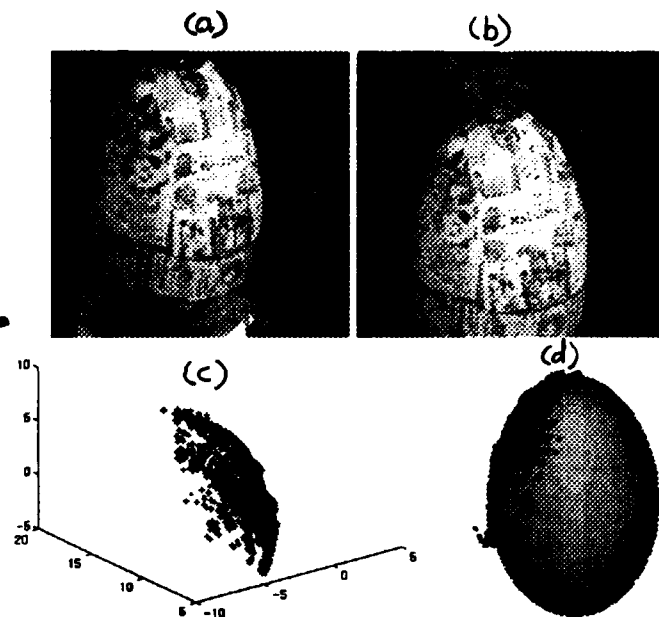


Figure 2: (a) and (b) : Stereo images of the object; (c) : Surface reconstruction using the stereo algorithm; (d) : Reconstructed 3D surface points superimposed on the database model for the object

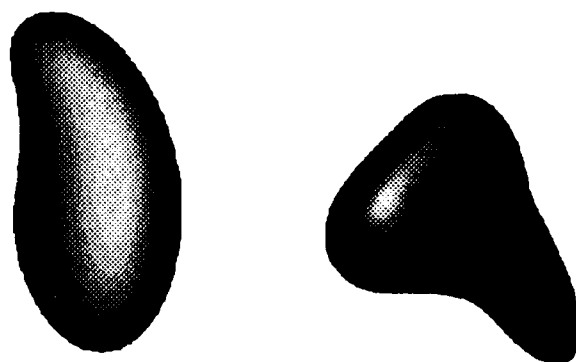


Figure 3: Examples of 4th degree polynomial surfaces fit to data sensed by robotic tracking around an eggplant and a pear, respectively, along parallel slices.

GMU RESEARCH ON LEARNING IN VISION: Initial Results

R. Michalski, J. Bala and P. Pachowicz
Center for Artificial Intelligence
George Mason University
Fairfax, VA 22030

Abstract

This report covers initial research on learning in vision conducted at the GMU Center for Artificial Intelligence. The research is currently conducted by two faculty members and one research assistant. The report describes our research goals, general approach, several developed methods, and results from experiments with implemented systems. The research has been concerned primarily with the development of efficient methods for inductive learning of texture descriptions from texture samples. The following methods (and implemented systems) are briefly described: *Textral* (employing multilevel symbolic image transformations and the AQ15 inductive learning program), *PRAX* (using a "principal axes" representation of texture descriptions), AQ-NT (oriented toward learning from noisy inputs), AQ-GA (combining inductive rule learning with a genetic algorithm based rule enhancement), and Chameleon (based on "model evolution" approach).

1 Introduction

The goal of this research is to explore the applicability of machine learning methods to problems of computer vision. The underlying premise is that computer vision will ultimately need to exhibit learning capabilities in order to be fully successful.

This research was supported in part by the Defense Advanced Research Projects Agency under the grant No. F49620-92-J-0549, administered by the Air Force Office of Scientific Research, and the grant No. N00014-91-J-1854, administered by the Office of Naval Research, in part by the Office of Naval Research under the grant No. N00014-91-J-1351, and in part by the National Science Foundation under the grant No. IRI-9020266.

The reasons for this view are based on the following observations:

- The world changes in unpredictable ways, therefore it is impossible, in principle, to pre-program in the vision systems all the knowledge necessary for image understanding.
- Handcrafting the knowledge needed for image understanding into computer vision systems is a difficult and time-consuming process; learning provides a fundamental vehicle for simplifying this process.
- In biological vision systems, many aspects of image perception are genetically preprogrammed, but many are learned. Similarly, computer vision systems should be able to acquire some capabilities through learning.

An important result of our initial research is a demonstration that symbolic learning methods can be successfully applied to selected problems of low-level vision, in which nonsymbolic methods have been traditionally employed. Specifically, the results obtained demonstrate that these methods have been very useful for creating descriptions of textures from their samples, obtained from the original camera-generated images.

2 General Approach

The developed approach, called "Multilevel Logical Templates" (MLT) aims at automatically determining texture class descriptions ("texture signatures") from texture samples. The basic step in this process is an iterative (multilevel) application of symbolic inductive learning to generate texture rules. These rules serve as "logical templates" that are

matched against window-size samples of texture classes.

The approach was originally proposed by Michalski [1973], and initially applied using the ILLIAC III image recognition computer facilities.

The research at the GMU Center for Artificial Intelligence has developed a variety of novel extensions and new directions stemming from the above general approach. The novelty is in utilizing new types of image transformations, self-improvement of the representation space (constructive induction), advanced noise-tolerant learning techniques, and new multistrategy learning techniques.

The basic idea behind the MLT approach can be explained as follows (Figure 1). Given an image with labeled samples of different textures, the learning system determines a sequence of operators that transform this image to a

“symbolic” image, in which picture elements are labels of corresponding texture areas.

The sequence of operators that produces such a labeling serves as a texture description (“texture signature”). The basic operator in this process is an application of a set of logic-style rules to transformed texture samples. The rules can be applied in parallel, and serve as “logical templates” that are applied to “events” (attribute vectors) representing texture samples.

To recognize an unknown texture sample, the system matches it with all candidate texture descriptions. This is done by applying decision rules to the events in the sample. For each event, the class membership (texture class) is determined.

The assignment of the sample to a given decision class (texture) is based on determining which of the candidate classes gets the majority (or) plurality of votes. Thus, even if some events in the sample are incorrectly recognized, the classification of the sample may be correct.

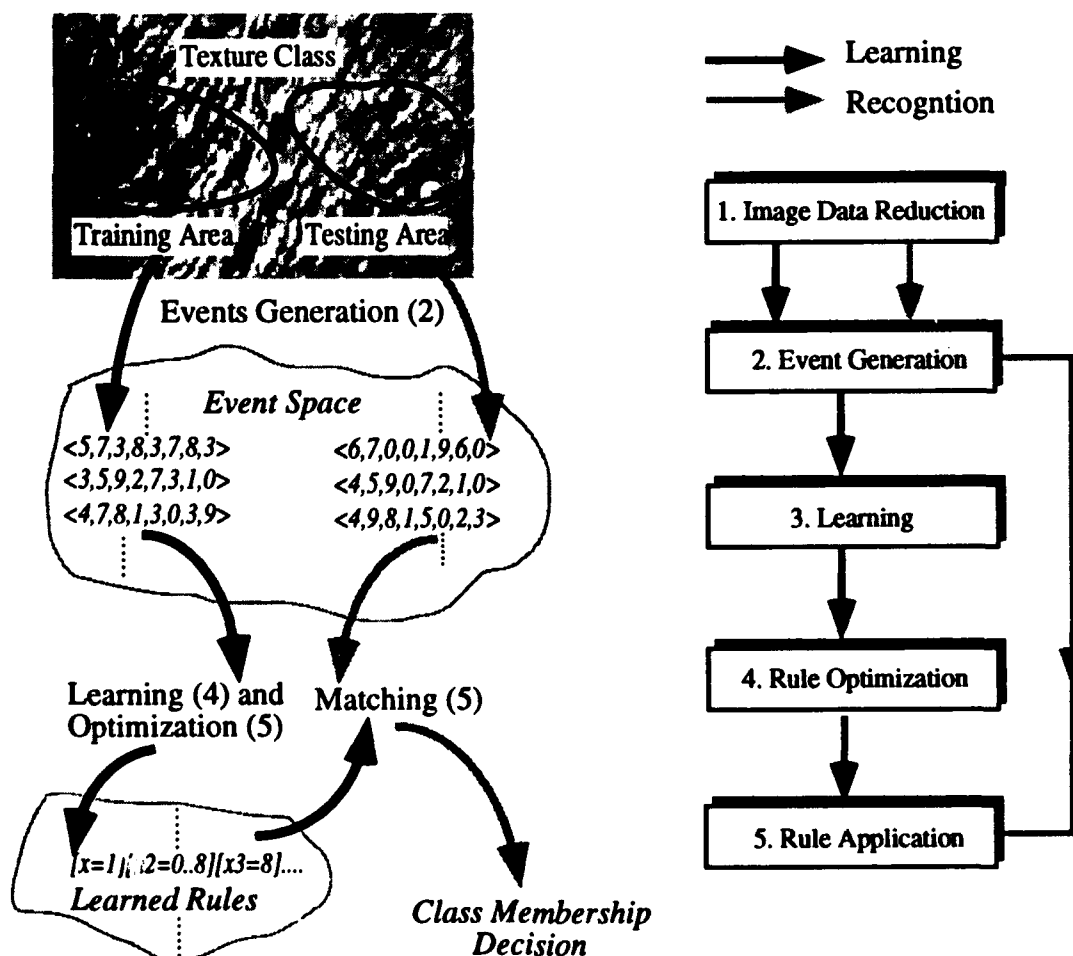


Figure 1: An illustration of the MLT approach to texture learning and recognition.

The process of learning such texture descriptions consists of the following phases (Figure 1):

- 1) Image preprocessing (volume reduction).
- (2) Training events generation (selection of texture samples, determining attributes, and formulating training examples)
- (3) Inductive learning of texture rules ("logical templates"), and
- (4) Texture rule optimization.

The above process may be repeated iteratively until a desired image transformation is obtained.

The first phase of the process adapts the "image volume" to the texture classes characterized by training samples.

This is done by modifying the spatial resolution and a gray-level resolution of the image so that the similarities between samples of the same texture and dissimilarities between samples of different textures are increased. In the initial experiments, the events were extracted from the second or third level of the Gaussian pyramid. The typical resolution of camera-acquired images was 512 by 512 image elements.

The second phase extracts a set of spatial texture samples, called *events*, from classified texture regions (Module 2 in Figure 1). An event is a vector of attribute values that represent different image (texture) features. Initial attributes are predefined. Additional attributes can be determined through the process of constructive induction [Wnek and Michalski, 1991].

There are many possible attributes that could be determined to characterize textures. The most desirable are those that define a description space in which points corresponding to the same texture class constitute easily describable clusters.

The attributes generated by different systems described in this report fall into one of three categories: neighboring gray-level values, statistical measurements, and convolution filter outputs. Sets of events extracted from texture classes to be learned are used as training examples.

Texture rules are determined using the AQ-15 method for inductive concept learning from examples ([Michalski, 1986]). The rules learned by the AQ method are represented in VL₁ (Variable-Valued Logic System 1); [Michalski, 1972]). Advantages of this representation are that it is amenable for parallel execution and easy to interpret conceptually.

In the machine learning method described here,, a concept corresponds to a single texture class. A concept description is a logical expression in disjunctive normal form associated with a decision class (here, a texture class).

Each conjunction in this expression together with the associated decision class can be viewed as a single decision rule.

The conjunctions (serving as condition parts of the rules) are logical products of elementary conditions in the form:

[L # R]

where:

L, called the *referee*, denotes an attribute.

R, called the *referent*, is a subset of values from the domain of the attribute L.

is one of the following relational symbols:

=, <, >, >=, <=, <>.

Each rule is assigned two parameters: "t" (for "total weight")—measuring the total number of positive training examples covered by the rule, and "u" (for "unique weight")—measuring the number of positive examples covered by the given rule and not covered by any other rule for the given decision class.

Here is an example of an AQ-15 decision rule:

[Class=1] \Leftarrow [x2=1][x4>3][x6=1..7]: (t=6, u=2)

This rule covers 6 examples of Class 1, out of which 2 are covered only by this rule, and not by any other rule for this class. In the case of texture rules, x_i are attributes characterizing a texture sample (in our experiments we used primarily 8x8 windows). The above rule is satisfied, if attribute x2 takes value 1, attribute x4 has value greater than 3, and attribute x6 takes value between 1 and 7.

As mentioned earlier, a description of a texture class can be viewed a set of such rules (a "ruleset"). In such a ruleset, individual rules are ordered according to the decreasing values of the t-weight. The following is an example of a texture description:

[Texture class = sweater surface] \Leftarrow
 [x1=7,9,12]& [x2>]&[x3=0..4]&[x4=0..5]&[x5=0..3]
 [x6=0..7] [x7=2..4] [x8=0..3] (t:28, u:21)
 OR
 [x1=5,7,9] [x2>2] [x3=0..2] [x4=0..4] [x5=1..4]
 [x6=0..6] [x7=0..2] [x8=0..4] (t:27, u:20)
 OR
 [x1=2,5,7]&[x2=1..12]&[x3=1..2]
 [x4=2..6]&[x5=3..4]&[x6=0..4]&[x7=1..3,5,7]
 [x8=0..1,3..4] (t:16, u:11)
 OR
 [x1=5..14]&[x2>6]&[x3=0..2,4..5]&[x4=2..5] (t:5, u:3)

where x1 is the Laplacian edge operator, x2 is the Frequency spot, x3 is the horizontal edge

operator, x4 is the vertical edge operator, x5 is the horizontal V-shape operator, x6 is the vertical V-shape operator, x7 is the vertical line operator, and x8 is the horizontal line operator.

The method uses "truncated" descriptions of texture classes. A truncated description is obtained by the removing from the initially generated rules the ones with a very low t-weight. The reason for this is that rules with a low t-weight can be viewed as insignificant, or as representing noise.

We have discovered experimentally that so truncated descriptions often give a higher texture recognition performance than non-truncated descriptions. Since truncated descriptions are also simpler, then such a truncation process is highly desirable. A detailed study of this phenomenon (in the context of non-vision applications) have been described in [Bergadano et al., 1992].

The learned texture descriptions are generalizations of the observed texture events (i.e., attribute-value vectors characterizing window-size texture samples). Therefore, they can be used to classify unobserved texture samples. There are two methods for applying the descriptions for recognizing the class membership of an event: the *strict* match and the *flexible* match.

In the strict match, the system tests whether an event strictly satisfies (the condition part of) a rule. The satisfied rule determines the classification decision. In the flexible match, the system computes a degree of match between the event and candidate rules. The degree of match can vary in the range from 0. (no match) to 1.0 (complete match). The rule with the highest degree of match determines the classification decision.

To explain the calculation of the degree of match, assume that a recognition rule contains a condition $[x = a_k]$. If the domain of the attribute x is a set of numerical values $\langle a_1, a_2, \dots, a_n \rangle$, and an event includes the statement $[x = a_j]$, the normalized degree of match between the rule and the condition in the event is defined:

$$1 - (|a_j - a_k| / n)$$

If the condition has several values in the referent (on its right-hand-side), the value closest to a_k is used. The degree of match between a rule containing several conditions and an event was computed as the average of the degrees of match between the conditions and the event conditions.

The degree of match between a class description (which may have several rules) and a given testing event (an example) is determined as the

maximum of the degrees of match between individual rules in the description and the event. The description with the highest match among classes determines the recognition decision. The measure of recognition accuracy of a rule when applied to a set of testing events is the percentage of the number of correctly classified test events to the total number of testing events in the set.

3 Implemented Systems

3.1 Learning Texture Signatures: TEXTRAL

The TEXTRAL system implements a version of the MLT approach ("Multiple Logical Templates"). The system generates multiple level of descriptions (rulesets) by applying the same learning process to images generated at each level. The first level ruleset relates to the original camera-acquired image. The next level ruleset relates to a "symbolic image" that consists of numerical labels associated with the texture classes. These labels are generated by the application of the first level ruleset, and represent texture classes assigned to texture events in the original image [Bala and Michalski, 1991]. Subsequent levels of rulesets are generated by reapplying this same process to the symbolic images generated at the previous step..

Here is a more detailed description of the algorithm:

- Step 1 extracts a random set of training events from the training areas in the original images by applying various local operators (such as Law masks, statistical measures, convolution operators, etc.), and learns the "first-level" texture of rules;
- Step 2 determines rulesets generalizing the training events. These rulesets are applied to the training areas of the original image, and a new image (a "symbolic image") is created. The pixels of the new image (the next level image) are numerical labels of texture classes assigned by the ruleset to corresponding events in the original (previous level) image.
- Step 3 determines the match between the texture training areas labeled by the teacher and the corresponding areas in the symbolic image. If the match is sufficiently high (or the system reaches a designated number of levels) then the process stops. Otherwise, the control is passed to the step 1. The events are extracted from the symbolic image (the last level image) and assigned classes corresponding to the training assignment of pixels in the original image (i.e., representing the "correct" partitioning of the image into texture classes done by the teacher).

We have performed a number of experiments with the system for various numbers of texture classes (between 4 and 16), representing fined-grained textures, such as sand, paper, pebbles, etc. Training events were determined from texture samplings using 8x8 windows, and selected from texture training areas. The texture training and testing areas for each texture class was determined by a teacher.

Table 1 and 2 show the confusion matrices characterizing the system's recognition rates (in %) for individual texture events (using 8x8 windows) selected from testing areas of four texture classes, C1, C2, C3 and C4. Table 1 shows the recognition rate for first level rules, and Table 2—for the second level rules. Recall that the conditions of the second level rules apply not to properties of the original image, but to the distribution of texture labels generated by the first level rules.

Recognized texture class				
Correct Class	C 1	C 2	C 3	C 4
C 1	84	15	16	23
C 2	10	78	20	10
C 3	7	14	79	27
C 4	26	17	27	67

Recognition rates using the first level rules.
Table 1.

Recognized texture class				
Correct Class	C 1	C 2	C 3	C 4
C 1	94	3	2	6
C 2	4	96	4	4
C 3	4	9	88	12
C 4	13	7	12	80

Recognition rates using the second level rules.
Table 2.

The average correct recognition rate of individual events for the 4 class experiment was 77% when using the first level rules, and 89.5% when using the second level rules. At the same time, the average misclassification rate decreased from 17.6% to 6.6%, respectively. Thus, the

experiment has demonstrated that multilevel learning (using higher level rules) can increase the system's recognition of individual events.

It should be clearly noted, however, that to recognize a given *sample* of a texture, one would extract from the unknown texture not just single event (representing an 8x8 window), but also several neighboring events. In such a case, the texture identification decision will be based on the majority of class assignments of individual events in the neighborhood.

Therefore, *even when there is a relatively low recognition rate of individual events, one can achieve 100% recognition rate of the sample* (it is sufficient that the plurality of events in the sample are recognized correctly). A problem may occur mainly when a sample is taken from a border area between different textures, or includes events characterizing rare local texture distortions.

Figure 2 presents the recognition rate of *individual events* on learning of 12 textures, using rules of level 1, 2, 3 and 4.

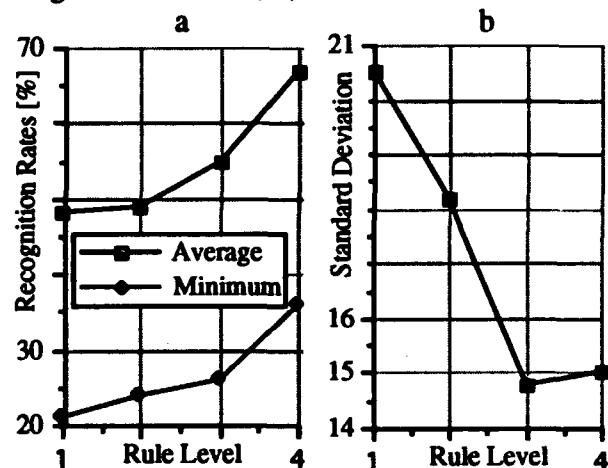


Figure 2: An increase of the system performance with the rule level.

Figure 2A shows the increase of the recognition rate of individual events from 12 texture classes with the rule level. Figure 2B shows the corresponding decrease of the standard deviation (in %) of the recognition rates with the rule level. The average recognition rate of individual events increased from 48% with level one rules to 58% with level four rules. At the same time, standard deviation (of correct recognition values) decreased from above 20 to 15, respectively. The minimum recognition rate increased from 21% to 36%.

The TEXTRAL method represents an extension and improvement of our earlier method implemented in the TEXPRT system [Channic, 1989]. TEXPRT used our earlier inductive learning system GEM ("Generalization of Examples by Machine;" also called AQ14) [Reinke, 1984]. TEXPRT was applied to the problem of recognizing faults in laminated aircraft materials using ultra-sound images

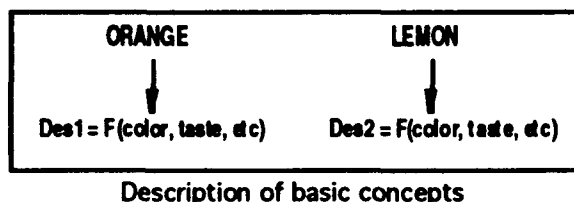
3.2 Learning Large Number of Classes: PRAX

In the TEXTRAL system, each texture class is represented by a ruleset [Bala et al., 1992]. If there are very many texture classes, there will be correspondingly many rulesets, and the learning and recognition process may become complex. The PRAX system represents an alternative approach to the problem of learning a large number of concept descriptions (in our application, texture classes).

The basic idea is to designate some concepts to be basic, and describe the remaining concepts in terms of the relations to the basic concepts. This idea can be simply illustrated by the example in Figure 3.

If the system already knows the concept of "orange" (Des1) and "lemon" (Des2), then it can learn the concept of "grapefruit" by relating properties of the grapefruit to those of the lemon and the orange (Des3'), rather than in terms of original properties (Des3'').

Phase I. Learning Basic Concepts



Phase II. Learning New Concept

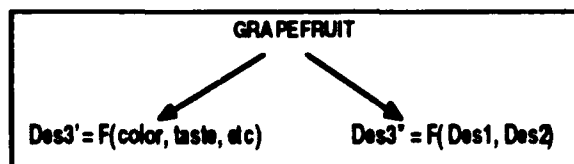


Figure 3: A simple illustration of the PRAX method.

In the PRAX method, descriptions of the basic concepts are called "principal axes." They are learned in the similar way as in the TEXTRAL system.

To learn a new, non-basic concept, the system determines a *similarity matrix* (SM) for that concept. The SM specifies the average degrees of similarity between the training examples of the new concept and all the principal axes.

The degree of similarity between an event and each principal axis is determined according to a procedure called ATEST [Michalski, et al, 1986]. The procedure determines the accumulated difference between the attribute values in the event and the conditions in each rule in the principal axis. To obtain a uniform representation of all class descriptions, the similarity matrix is also computed for all basic concepts.

These degrees of similarity can be viewed as values of the new constructed attributes. Thus, this method represents a special case of constructive induction. (The general concept of constructive induction includes any method that self-modifies the concept representation space during the induction process. Generating additional, problem oriented attributes is an important form of such self-modification of the representation space [Michalski, 1978; Wnek & Michalski, 1991]).

To recognize an unclassified event, the method creates an SM for it, that is, determines a matrix of similarities between the event and the principal axes. Subsequently, the system determines the best match between the SM of that event and SMs of all candidate concepts. The best match indicates the class membership.

The method was empirically evaluated by applying it to the problem of learning 24 texture classes from examples (Table 3). Each example was described in terms of eight multivalued attributes (representing detectors of various basic geometrical concepts, such as the presence of lines, edges, V-shapes, etc.). The performance of the PRAX-derived descriptions was compared with the performance of the k-NN classifier. Different level of misclassification noise were added to test the robustness of the method.

The main strength of the method lies in a problem-relevant transformation of the descriptor space. The new descriptors form generalized sub-spaces of the initial, training space. In addition, the method uses a non-linear distance metric to calculate values of constructed attributes. The distance metric based on the idea of flexible matching is less sensitive to noise, then traditional Euclidean distance metric often used by pattern recognition methods.

METHOD	The Recognition Rate (in %) of Examples from Unknown Texture
PRAX	
No Noise	100%
5% Noise	100%
10% Noise	100%
K-NN	
No Noise	96%
5% Noise	92%
10% Noise	87%

Table 3: The results from comparing PRAX with the K-NN method.

The current problem with the method is that it does not have a mechanism for deciding how to choose basic concepts. Choosing the minimal subset of concepts to be used for principal axes generation is important for method to be efficient. This problem will be a subject of future research. Another weakness is that the similarity matrix is a relatively complex representation.

3.3 Learning From Noisy Data: AQ-NT

The AQ-NT method represents a novel way of handling problems of learning from noisy real-world data [Pachowicz and Bala, 1991]. It is based on the idea that events covered by rules with a low t-weight may be representing noise in the data. The assumption is that the system learning from a dataset that does not contain such events has a greater chance to produce correct concept descriptions than when learning from the original events.

The process of learning concept descriptions (in the form of a ruleset) is done in the following two phases:

Phase 1: Performs a rule-based "filtration" of the noise from the training data. This is done in the following way:

1. Induce decision rules from a given dataset using the AQ learning program.
2. Truncate concept descriptions by removing "the least significant" rules, defined as rules that cover only a small portion of the training data (have small t-weight relative to the t-weight of other rules).
3. Create a new training dataset that includes only training examples covered by the modified concept descriptions.

4. If the size of the dataset falls below an assumed percentage of the training data (which reflects an assumed error rate in the data), then go to Phase 2. Otherwise, return to step 1.

Phase 2: Acquire concept descriptions from the reduced training dataset using the AQ learning program.

Figure 4 shows results from one run of the AQ-NT system for six texture classes.

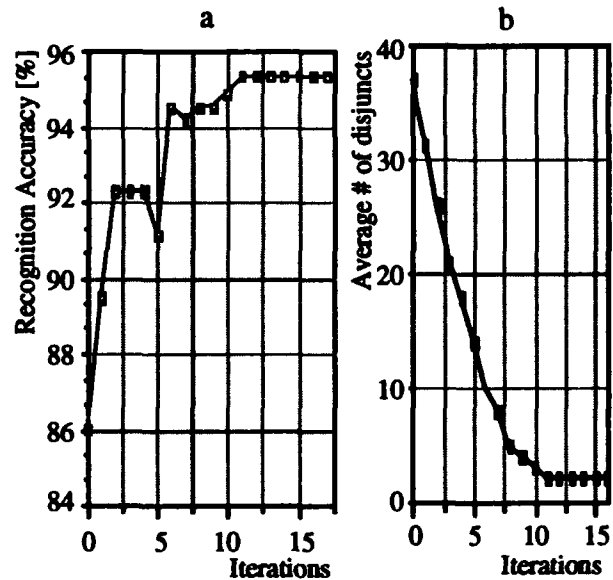


Figure 4: The AQ-NT results.

Figure 4A shows the increase of the recognition accuracy (in %) of individual events with the number of iterations. After 12 iterations, the recognition accuracy reached 95.3%. Figure 4B shows the average number of rules for each iteration. An average number of rules can be viewed as a measure of description complexity. Figure 4B shows a significant decrease in the average number of rules (from 37 to 3). This result is a significant indication of the advantages of the proposed approach.

3.4 Rule Improvement by Genetic Algorithm: AQ-GA

The size, complexity, variability and an inherent noise in the vision data pose significant difficulties in developing a reliable concept learning system. The AQ-GA multistrategy system was developed to address some of these issues [Bala et. al., 1993]. This system integrates two forms of learning, symbolic inductive generalization and genetic algorithm based learning. The integration is done in a closed-

loop fashion in order to achieve robust concept learning capabilities.

The learning process cycles through two phases (Figure 5).

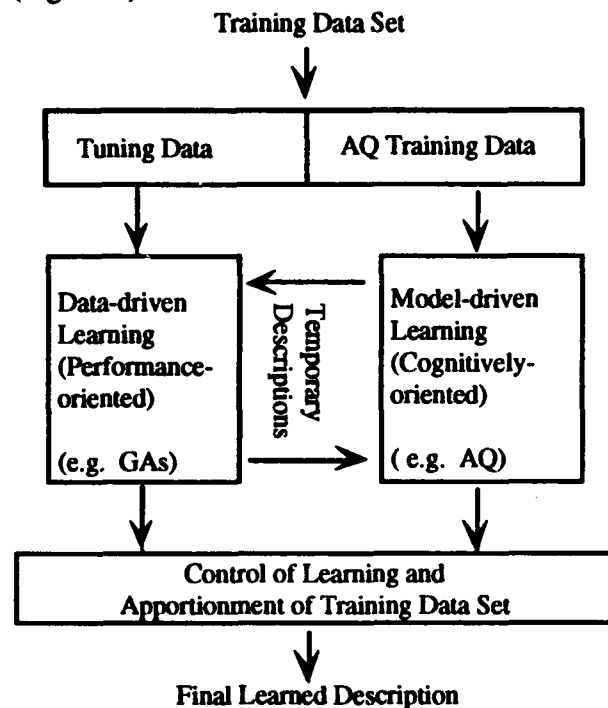


Figure 5: The AQ-GA architecture.

In the first phase, initial concept descriptions are acquired by running a noise-tolerant extension of the AQ15 rule induction system. The resulting concept descriptions may not be optimal from the performance viewpoint, due to the AQ bias to generate simple, cognitively-oriented descriptions. Therefore, in the second phase, the system attempts to improve the performance of the descriptions by employing a genetic algorithm (GA).

The descriptions obtained from AQ15 are semi-randomly modified, using basic genetic operators: mutation and crossover. The resulting descriptions are evaluated according to a performance criterion. The criterion was the recognition accuracy of the descriptions on the "tuning" data (a subset of the training set of events). The best performing descriptions are selected from the population, and a new generation is repeated. The process stops when a desirable performance level is achieved, or the number of generations exceeds some limit.

The effectiveness of this multistrategy approach was tested on several texture recognition problems.

Genetic algorithms typically represent individuals in a population (here, concept descriptions), using fixed-length binary strings. However, if the effective cooperative learning A novelty of this method is that it uses, instead of binary strings, concept descriptions (formally, VL₁ expressions) produced by AQ15. To this end, a special mutation operator was designed to introduce small changes to selected condition parts of the rules in each concept description. The condition parts are selected by randomly generating two pointers: the first selects a rule, and the second one selects a condition in this rule.

The most-left or the most-right values of the referent in this condition are slightly modified.. For example, the condition $[x1 = 10..23]$ might be mutated to any of the following: $[x1 = 10..20]$, $[x1 = 10..24]$, $[x1 = 12..23]$ or $[x1 = 8..23]$, as well as others. Such a mutation process samples the space of possible concept description boundaries to improve the performance criteria. The mutation process can be viewed as equivalent various *transmutations* (knowledge transformations; Michalski, 1993) of the conditional part of a rule.:

- specialization: $[x5 = 3, 10..23] \Rightarrow [x5 = 3, 10..20]$
- generalization: $[x5 = 3, 10..23] \Rightarrow [x5 = 3, 10..24]$
- variation: $[x5 = 3, 10..23] \Rightarrow [x5 = 5, 10..23]$

The crossover operation is performed by splitting concept description into two parts, upper rules and lower rules. These parts are exchanged between parent concept descriptions to produce new child concept descriptions. Since the degree of match of a given tuning event depends on the degree of match of this event to each rule of concept description, this exchange process enables inheritance of information about strong rules (strongly matching) in the individuals of the next evolved population. An example of crossover applied to short, four rules description is depicted below:

Parent description 1

```

1  [x1=7..8] [x2=8..19] [x3=8..13] [x5=4..54]
2  [x1=15..54] [x3=11..14] [x6=0..9] [x7=0..11]
-----crossover position-----
3  [x1=9..18] [x3=16..21] [x4=9..10]
4  [x1=10..14] [x3=13..16] [x4=14..54]
  
```

Parent description 2

```

1  [x1=16..54] [x5=0..6] [x7=5..12]
2  [x1=8..25] [x3=8..13] [x4=9..11] [x5=0..3]
-----crossover position-----
3  [x4=0..22] [x5=8..9] [x6=0..7] [x7=11..48]
4  [x2=5..8] [x3=7..8] [x4=8..11] [x5=0..3]
  
```

The result of the crossover operation (one of two child descriptions) is the following:

```

1 [x1=7..8] [x2=8..19] [x3=8..13] [x5=4..54]
2 [x1=15..54] [x3=11..14] [x6=0..9] [x7=0..11]
3 [x4=0..22] [x5=8..9] [x6=0..7] [x7=11..48]
4 [x2=5..8] [x3=7..8] [x4=8..11] [x5=0..3]

```

The performed experiments, involving learning rules for describing texture classes, demonstrated that the classification results obtained with the hybrid learning algorithm (Figure 5) (AQ Training Data -> AQ and Tuning Data -> GAs) exceed the performance of the AQ algorithm used alone (AQ Training Data + Tuning Data -> AQ). Figure 6 presents recognition rates from one of the experiments.

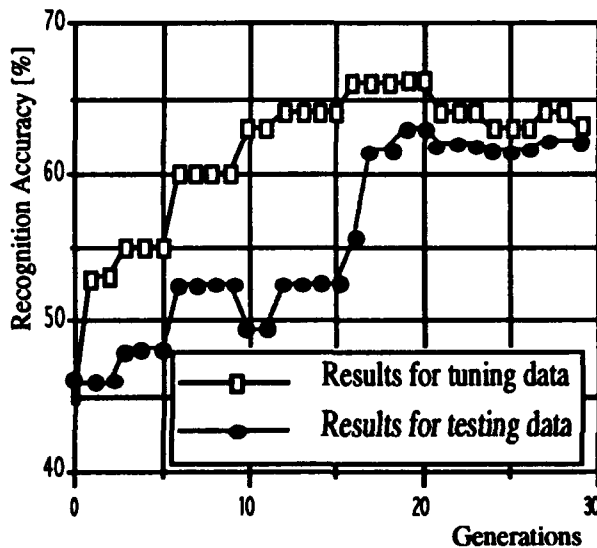


Figure 6: Recognition accuracy during the GA generations.

The result shows the recognition rates for the class that was optimized by the GA. White marks on the diagrams represent results obtained for tuning data and black marks represent recognition results for testing data.

3.5 Texture Recognition Under Varying Perceptual Conditions: Chameleon

In recognizing natural objects outdoors we have to deal with a great variability of perceptual conditions that influence changes in object visual characteristics. Most vision research on object recognition in outdoor environments, however, has been focused on recognizing objects under stationary conditions rather than dynamic conditions (varying resolution, lighting, pose, and state). Object models, particularly texture models, when learned under a given

conditions are not effective in recognizing objects under different perceptual conditions.

To develop robust object recognition systems, we have to implement system adaptation capabilities that can mitigate influence of changing perceptual conditions on the effectiveness of object recognition. Our ultimate goal is to integrate learning and vision modules in such a way that learning functions can support adaptation functions of the vision system.

The variability of texture characteristics under changing resolution, lighting and pose has been investigated. It was found that texture attribute distribution (for different attribute extraction methods) can vary significantly when these conditions are changed. The shape of the distribution often contains a multimodality of texture characteristics. In most cases studied, the distribution cannot be determined *a-priori*. We also observe that the variability of perceptual conditions causes a significant translation of the attribute distribution within the attribute space.

Relatively little has been done on the application of machine learning methods to the adaptability of vision systems to the dynamic environment. Bhanu et al. [1989, 1990] apply genetic algorithms to image segmentation problems with an extension towards segmentation under variable perceptual conditions.

The variability of object appearance (particularly texture characteristics) requires the development of system capabilities that will dynamically reconfigure and update object models (knowledge). In our approach [Pachowicz, 1991], system adaptation is applied to recognize objects on images acquired over time.

In order to recognize an object on images sequentially, the system has to iteratively update the object model with regard to changes in object characteristics. In this approach, a time sequence of images monitors slight changes in resolution, lighting and surface positioning from one image to the next one -- a sequence of images is affected by continuous changes in perceptual conditions.

We integrate the learning and recognition processes within a closed loop to update texture models. Analysis of system recognition effectiveness, performed over a sequence of

images, detects changes in textures. If this effectiveness decreases then the system activates incremental learning processes of model modification to improve the model discriminating power. The system learns initial texture models from teacher-provided examples. Then, the system updates these descriptions automatically without teacher help.

Two systems were developed: CHAMELEON '91 and CHAMELEON '92. The first system had only some of the adaptability functions implemented [Pachowicz et al., 1992, Pachowicz, 1992]. In this system, a teacher segments each image in a sequence. The system was useful for investigating stability problems and for the modification of object models performed on-line. The second system is more autonomous, and needs much less help from a human operator. The underlying methodology, system architecture, and experimental results are presented in a separate paper in the Proceedings [Pachowicz, 1993].

4 Summary

We have presented a general approach, called Multilevel Logical Templates (MLT), and several implemented systems for inductive learning descriptions of texture classes from texture samples. These systems represent different variations and extensions of the general approach oriented toward various types of learning problems:

- **TEXTRAL** —for multilevel learning to maximally improve the recognition accuracy of new textures,
- **AQ-NT**—for learning from data with noise,
- **PRAX** — for learning from large numbers of texture classes,
- **AQ-GA** — for automatic tuning and enhancement of concept descriptions obtained by a standard AQ inductive learning method,
- **Chameleon** — for model evolution under changing perceptual conditions.

The systems have been tested on several texture recognition problems, and the results were very encouraging. The recognition rates for even relatively high number textures (36) were frequently near 100%

The developed methodology is quite general and can be potentially applied not only to the

problem of learning of the texture descriptions, but also to other types of problems in vision.

There are several limitations of the current methods. We have not investigated issues of the robustness and the sensitivity of the methods to various invariant texture transformations (e.g., a significant changes in the illumination). Also, it is unclear how the performance of the methods depends on the number of texture classes.

There are several other major topics to be investigated in future research:

- (i) the enhancements to the current learning methodology to include capabilities for automatically generating higher level problem-relevant attributers (constructive induction)
- (ii) the applicability of multistrategy learning (e.g., combining symbolic rule learning with neural network learning; the issue of representing and learning of imprecisely defined visual concepts).
- (iii) the extensions of the methodology to other problems in vision, e.g., learning of shape classes.
- (iv) learning new visual concepts in terms of differences and similarities from known concepts, and developing a calculus for representing symbolic differences between visual concepts.

Acknowledgments

The GMU research on machine learning and vision is conducted in collaboration with the Center for Automation Research at the University of Maryland.

Authors wish to thank Eric Bloedorn for his contribution to the development of the PRAX system, and for constructive comments about the paper. They also thank Ken De Jong for his collaboration in the development the AQ-GA system.

References

- Bala, J., DeJong K., and Pachowicz P., "Multistrategy Learning From Engineering Data by Integrating Inductive Generalization and Genetic Algorithms," in *Machine Learning: A Multistrategy Approach Volume IV*, R.S. Michalski and G. Tecuci, (Eds.), Morgan Kaufman, San Mateo CA., 1993 (in press).

Bala, J. and Pachowicz P., "Recognizing Noisy Pattern Via Iterative Optimization and Matching of Their Rule Description," *International Journal on Pattern Recognition and Artificial Intelligence*, vol. 9, issue 4, pp. 513-538, 1992.

Bala, J., Michalski R.S., and Wnek J., "The Principal Axes Method for Constructive Induction," *Proc. of the 9th International Conference on Machine Learning*, Aberdeen, Scotland, July 1992.

Bala, J. and Michalski R., "Recognition of Textural Concepts Through Multilevel Symbolic Transformations," *Proc. of the 3rd International Conference on Tools for Artificial Intelligence*, San Jose C.A., Nov. 1991.

Bergadano, F., Matwin, S., Michalski, R. S., & Zhang, J., "Learning Two-tiered Descriptions of Flexible Concepts: The POSEIDON System," *Machine Learning*, No. 8, 5-43, 1992.

Bhanu, B., S. Lee and J. Ming, "Adaptive Image Segmentation Using a Genetic Algorithm," *Proc. of Image Understanding Workshop*, Palo Alto, CA, pp.1043-1055, 1989.

Bhanu, B., S. Lee and J. Ming, "Self-Optimizing Control System for Adaptive Image Segmentation," *Proc. of Image Understanding Workshop*, Pittsburgh, PA, pp.583-596, 1990.

Channic, T., "TEXPERT: An Application of Machine Learning to Texture Recognition," A publication of the Machine Learning and Inference Laboratory; MLI 89-17, George Mason University, Fairfax, Virginia.

Michalski, R. S., "A Variable-Valued Logic System as Applied to Picture Description and Recognition," in *Graphic Languages*, Nake, F. and Rosenfield, A. [Eds.], North Holland, 1972.

Michalski R. S., "AQVAL/1--Computer Implementation of a Variable-Valued Logic System VL1 and Examples of Its Application to Pattern Recognition," in *Proc. of the 1st International Joint Conference on Pattern Recognition*, October 30, 1973, Washington DC.

Michalski, R.S., "Pattern Recognition as Knowledge-Guided Computer Induction," Report No. 927, Department of Computer Science, University of Illinois, Urbana, June 1978. (Subsequently published under the title "Pattern Recognition as Rule-Guided Inductive Inference," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, No. 4, pp. 349-361, July 1980.)

Michalski, R. S., "Inferential Theory of Learning: Developing Foundations for Multistrategy Learning," in *Machine Learning: A Multistrategy Approach Volume 4*, R.S. Michalski and G. Tecuci (Eds.), Morgan Kaufman Publishers, San Mateo, CA, 1993.

Michalski R.S., Mozetic, I., Hong, J., and Lavrac, N., "The multipurpose incremental learning system AQ15 and its testing application to three medical domains," *Proc. of the 5th National Conference on Artificial Intelligence*, 1986.

Pachowicz, P.W., "Invariant Object Recognition: A Model Evolution Approach," in these Proceedings, 1993.

Pachowicz, P.W., "A Learning-Based Evolution of Concept Descriptions for an Adaptive Object Recognition," *Proc. of the IEEE of Int. Conf. Tools with AI*, Arlington, pp.316-323, 1992.

Pachowicz, P.W., "Recognizing and Incrementally Evolving Texture Concepts in Dynamic Environments: An incremental model generalization approach," Report MLI-91-10, Artificial Intelligence Center, George Mason University, 1991.

Pachowicz, P. and Bala J., "Improving Recognition Effectiveness of Noisy Texture Concepts Through Optimization of Their Descriptions", *Proceedings of the 8th International Workshop on Machine Learning*, Evanston, Ill, June, 1991.

Pachowicz, P.W., M. Hieb and P. Mohta, "A Learning-Based Incremental Model Evolution for Invariant Object Recognition," *Proc. of the 6th International Conference on System Research, Informatics and Cybernetics*, Baden-Baden, August 1992.

Reinke, R.E., "Knowledge Acquisition and Refinement Tools for the ADVICE META-EXPERT System," ISG 84-4, UIUCDCS-F-84-921, Department of Computer Science, University of Illinois, Urbana, 1984.

Wnek J. and Michalski, R.S., "Hypothesis-driven Constructive Induction in AQ17: A Method and Experiments," *Proc. of the IJCAI-91 Workshop on Evaluating and Changing Representations in Machine Learning*, K.Morik, F.Bergadano and W.Buntine (Eds.), pp.13-22, Sydney, Australia, 1991.

Image Understanding Research at Johns Hopkins

Lawrence B. Wolff
Computer Vision Laboratory
Department of Computer Science
The Johns Hopkins University
Baltimore, Maryland 21218

Abstract

We have made progress in a number of areas this past year, primarily in Physics-Based Vision. We have built and are continuing to develop different versions of a new type of sensor called a *polarization camera* which we believe will make the more general capabilities of polarisation vision more accessible to many image understanding applications. An advancement in the modeling of diffuse reflection from smooth dielectric surfaces has been made, resulting in a relatively simple formula that significantly generalizes Lambert's Law. We have developed a new stereo technique which utilizes photometric ratios as an invariant across a stereo pair of cameras for point correspondence on smooth surfaces, producing accurate dense depth maps. Shape recovery techniques have been studied which utilize both reflectance and range data to compute shape more accurately than from each individual data source alone.

1 Polarization Vision

There is a compelling motivation to study polarization vision - polarisation affords a more general description of light than does intensity, and can therefore provide a richer set of descriptive physical constraints for the interpretation of an imaged scene. As intensity is the linear sum of polarisation components, intensity images physically represent reduced polarisation information. Because the study of polarisation vision is more general than intensity vision there are polarisation cues that can immensely simplify some important visual tasks (e.g., material classification, reflection component analysis, identification of specular reflection, image region segmentations, etc...) which are more complicated or possibly infeasible when limited to using intensity and color information. A detailed description of a variety of polarisation-based vision methods are contained in [14], [15], [25], [16], [3].

1.1 Polarisation Camera Computational Sensors: The Big Picture

A criticism that has sometimes been leveled at polarisation-based vision methods is the inconvenience of obtaining polarisation component images by having to place a linear polarising filter in front of an intensity CCD camera and mechanically rotating this filter by hand or by motor into different orientations. This inconvenience is simply a result of commercially available camera sensors being geared towards taking intensity images instead of polarisation images. In our conception, polarisation vision is no more a "multiple view" problem than is color vision, and a camera can be developed that can automatically sense polarisation components and even automatically compute physical scene properties that are directly

related to this polarisation information. Such a *polarization camera* sensor was originally suggested in [16], and in the past year at Johns Hopkins we have built a liquid crystal implementation of a polarisation camera [26]. We are continuing to build a variety of other polarisation camera sensors including a beamsplitter implementation, and a self-contained VLSI chip implementation some of which is discussed in an article contained in these proceedings [23].

Because humans do not observe polarisation directly except with the aid of special filters, it is beneficial for a polarisation camera to produce some kind of visualisation for representing sensed polarisation information (e.g., intensity-color representation) for scene analysis. We utilize a hue-saturation-intensity visualisation for partially linearly polarised light [26], [23]. Such a scheme was suggested by Bernard and Wehner [1] as a functional similarity between polarization vision and color vision for biological vision systems. Whether a polarisation camera computes a visualisation of sensed polarisation information at each pixel, or computes a visualisation of physical information (e.g., dielectric/metal composition) at each pixel related to sensed polarisation, a polarisation camera is inherently a *computational sensor*. The speed at which such computations can be performed is important for real-time applications.

We feel that there are considerable advantages to building a polarisation camera sensor geared towards doing polarisation vision. There already exist polarisation-based vision methods that can significantly benefit a number of application areas such as aerial reconnaissance, autonomous navigation (e.g., UGV), target recognition, inspection, and, manufacturing and quality control. A polarisation camera would make polarisation-based vision methods more accessible to these application areas and others. It should be fully realized that, as intensity is a compression of polarisation component information, a polarisation camera can function as a conventional intensity camera, so that intensity vision methods can be implemented by such a camera either alone, or, together with polarisation-based vision methods. As intensity-based methods are physical instances of polarisation-based methods, a camera sensor geared towards polarisation vision does not in any way exclude intensity vision, it only generalizes it providing more physical input to an automated vision system! Adding color sensing capability to a polarisation camera makes it possible to sense the complete set of electromagnetic parameters of light incident on the camera.

We are also considering, once a high resolution VLSI polarisation camera chip has been eventually made, of using such a chip for *polarization goggles* extending human vision into the polarisation domain. This is in analogy

with the way night vision goggles extend human vision to "see" other wavelengths of light. Polarization goggles may be useful to a number of areas which require this type of enhanced vision. It is known that many biological animals (e.g., bees and fish) receive important visual information in the polarization domain.

A patent is pending for a variety of these polarization camera and polarization goggle devices discussed above [13].

1.2 Liquid Crystal Polarization Camera

A polarization camera has been designed and built in the Computer Vision Laboratory that utilizes two Twisted Nematic (TN) liquid crystals in series with a fixed polarizer analyzer placed in front of a standard intensity CCD camera. See Figure 1. The TN liquid crystals electro-optically rotate the plane of the polarization of light, controlled by electrical voltages placed across the liquid crystals in synchronization with camera video. Not only does this obviate the need for mechanically rotating a polarizing filter in front of a CCD camera to image polarization components, but optical distortion caused by the wobbling from such mechanically rotation is virtually eliminated. Components of polarization are imaged under full automatic computer control, and these are processed on a Datacube MV-20 board programmable via Image flow software from a SUN workstation. For details see [26], [23]. One program on the Datacube MV-20 computes from polarization component images a hue-saturation-intensity visualization at each pixel for partial linear polarization representing, respectively, the orientation of the plane of the linear component of polarization, partial polarization (i.e., percentage of linear polarization content), and, intensity. Another program computes dielectric/metal composition from polarization component images. Our liquid crystal polarization camera can generate up to 2.5 polarization images a second. The main timing bottleneck is the relaxation time of 100ms for each of the liquid crystals to switch states. With the most current faster liquid crystals we can at least double the rate of polarization images per second, and we intend to incorporate these newer liquid crystals in our implementation. A nice feature about our liquid crystal polarization camera is that with the Datacube MV-20 board, it is a programmable computational sensor in that sensed polarization components can be processed in a variety of ways.

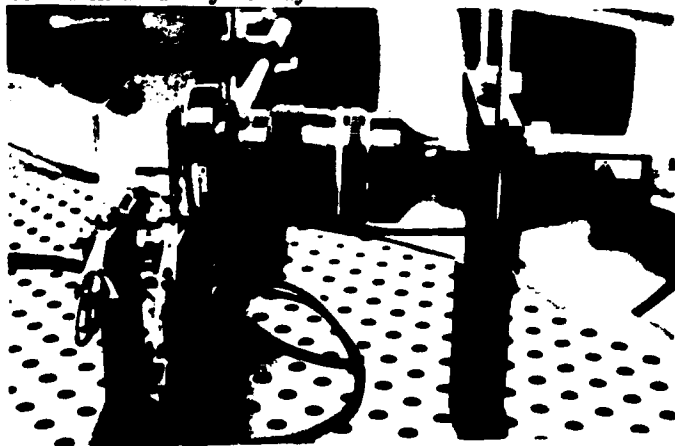


FIGURE 1

Depending upon interest from the image understanding community, a self-contained optical head can be made from liquid crystals and a polarizer, with appropriate electrical contacts, that can be mounted on the lens for a CCD

camera. Assuming the existence of hardware for digitizing and processing polarization component images this is a low cost way of converting an intensity CCD camera into a polarization camera.

1.3 Polarization Camera Using Beamsplitter

A common design for high quality color cameras is to use a beamsplitter that directs equal amounts of incoming light onto 3 separate CCD chips for red, green, and, blue. A similar idea can be used to direct light onto multiple CCD chips, each chip covered by a uniquely oriented polarizing filter. Unfortunately the polarizing properties of most common kinds of beamsplitters can be variable across standard wide fields of view.

We are developing a prototype for a 2-CCD polarization camera utilizing a polarizing plate beamsplitter. See Figure 2. The simplicity of this design stems from the use of a special coating on a glass plate producing a beamsplitter that effects the polarization of transmitted and reflected light in a nearly constant known way across a fairly wide range of angles (i.e., $\pm 20^\circ$). Some details are described in [23] contained in these proceedings. The polarization state of reflected and transmitted light is effected in a linearly independent way by the plate beamsplitter. If reflected and transmitted light are incident on different CCD chips, the horizontal and vertical components of polarization can be resolved by solving a linear set of simultaneous equations, and without the need for any polarizing filters on the CCD chips. The current trade-off between this type of polarization camera and our liquid crystal polarization camera is speed vs. amount of polarization information. While the 2-CCD polarization camera with beamsplitter can operate at least at 15 frames/second and probably at 30 frames/second, only two components of polarization are resolved as compared with the complete set of three components resolved by the liquid crystal polarization camera needed to compute a complete state of partial linear polarization. One way of extending the 2-CCD camera design to resolve three components of polarization is to add a TN liquid crystal intercepting light before it reaches the beamsplitter. There are obvious extensions using three CCD chips at the expense of more difficult registration problems. However, the 2-CCD polarization camera with polarizing plate beamsplitter currently appears to be a simple robust design that may be able in the short term to give near frame-rate, partial capability for polarization vision for applications such as the DARPA Unmanned Ground Vehicle.



FIGURE 2

1.4 Polarization Camera Chips

In collaboration with Prof. Andreas Andreou in the Electrical and Computer Engineering department at Johns Hopkins we are in the process of developing self-contained VLSI versions of polarization cameras that sense complete states of partial linear polarization on-chip. Currently we are experimenting with chips (designed at Hopkins and fabricated at MOSIS) that compute on-chip a hue-saturation-intensity representation of partial linear polarization at each "polarization pixel" and output this directly in video. VLSI offers very high computational throughput so that VLSI polarization cameras can potentially operate at very high speeds and benefit a number of application areas that require real-time operation.

1.5 Modeling of Polarization in Outdoor Scenes

Skylight is partially polarized, dependent upon position relative to the sun according to the law of Rayleigh scattering. Reflected polarization from different terrain types (e.g., water, vegetation, soil, etc...) is dependent upon which patch of skylight is producing the reflection. A polarization reflectance model is proposed in [19] which combines the polarization model originally presented in [14] with a polarization model for skylight illumination. Such a model can be used to predict reflected polarization dependent upon viewer orientation relative to surface terrain and could be potentially useful in providing insight into polarization-based identification of terrain type for autonomous navigation.

2 Reflectance Modeling

2.1 Diffuse Reflection

Perhaps the most widely used assumption about reflectance from materials in computer vision and in computer graphics is Lambert's Law for diffuse reflection [8]. Lambert predicted that diffuse reflection from a material contributed by light incident from a specified direction is proportional to the cosine of the angle between this incident direction and the surface normal, independent of the direction of reflection. While relatively little physical motivation was given for this law when it was first published over 200 years ago, it has been adopted by the computer vision and computer graphics communities primarily because it serves as a reasonably accurate and computationally simple approximation for describing diffuse reflection under a number of conditions.

A prevalent class of materials encountered both in common experience and in vision/robotics environments are inhomogeneous dielectrics which include plastics, ceramics, and, rubber. It has been known that diffuse reflection from smooth inhomogeneous dielectric surfaces can seriously deviate from Lambert's Law under certain conditions. We have formally derived from first physical principles and extensively empirically verified a relatively simple formula for diffuse reflection from smooth inhomogeneous dielectric surfaces that accurately explains striking deviations from Lambertian behavior [21], [18], [17]. Using the geometry depicted in Figure 3, if light is incident with radiance, L , at incidence angle, ψ , through a small solid angle, $d\omega$, on a smooth dielectric surface, then

$$\rho L \times (1 - F(\psi, n)) \times \cos \psi \times (1 - F(\sin^{-1}(\frac{\sin \phi}{n}), 1/n)) d\omega \quad (1)$$

describes the diffuse reflected radiance into emittance angle (i.e., viewing angle), ϕ . The terms F refer to the Fresnel reflection coefficients [11], n , is the index of refraction of the dielectric medium, and, ρ , is the *diffuse albedo*. This diffuse reflectance formula accurately describes the dependence of diffuse reflection on viewing angle, falling off to zero as viewing approaches grazing. This formula also accurately shows that diffuse reflection falls off faster than predicted by Lambert's law as a function of angle of incidence, particularly as angle of incidence approaches close to 90° . For exact details see [22] in these proceedings.

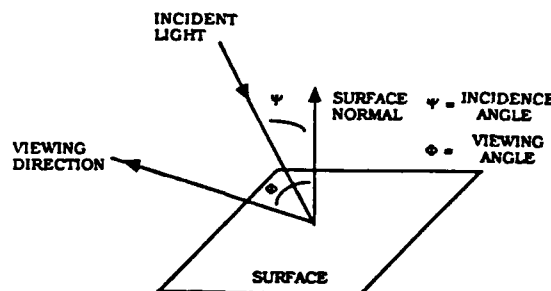


FIGURE 3

When the Fresnel reflection coefficients are close to zero, formula 1 becomes almost identical to Lambert's Law, and this is true to good approximation when incidence and emittance angles are both within the range of $0^\circ - 50^\circ$. This explains why Lambert's Law has generally been accepted as a reasonably good approximation. However, if either one or both incidence and emittance angles are outside this range, major deviations from Lambert's Law occur as the Fresnel coefficients become significant. In [22] (these proceedings) experiments are shown that illustrate striking non-Lambertian effects near occluding contours under oblique illumination that are accurately explained by our new formula.

Formula 1 has bearing on virtually any technique in computer vision and image understanding that relies upon the Lambertian assumption applied to dielectric surfaces, including shape from shading, shape and/or roughness determination from multiple light source illumination (e.g., photometric stereo) and shape from interreflection. It is impossible to reference all of the related works but the book by Horn and Brooks [6] contains a number of applicable papers. This result makes it possible to precisely analyze the conditions under which it is reasonable to assume the Lambertian model for a particular technique, and the conditions under which the Lambertian model breaks down. In turn our formula can be utilized to precisely analyze when certain image understanding techniques are valid. This more general diffuse reflectance model provides a more solid physical foundation upon which to develop accurate object feature extraction techniques in computer vision.

2.2 Relative Strength of Specular and Diffuse Reflection: How Bright is a Specularity?

An additional advancement that our diffuse reflectance model for inhomogeneous dielectrics makes is that it directly relates the diffuse albedo, ρ , to physical surface parameters. As explained in [21], [18], [17], [22] contained in these proceedings, inhomogeneous dielectric material is modeled as a collection of scatterers contained in a uniform dielectric medium with index of refraction different

from that of air. Diffuse reflected intensity results from the process of incident light refracting into the dielectric medium, producing a subsurface diffuse intensity distribution from multiple internal scattering, and then refraction of this subsurface diffuse intensity distribution back out into air. We show that the diffuse albedo, ρ , is dependent upon both the *single scattering albedo*, ρ , describing the proportion of energy reradiated upon each subsurface single scattering, and, the index of refraction n . The exact relationship is given in [18], [17], [22].

Whereas before diffuse albedo was just an *ad hoc* scaling coefficient our diffuse reflection model explains the physical origin of diffuse albedo. This means that combined diffuse and specular reflection can be modeled purely in terms of physical material parameters [20], [22]. An important consequence of this is that the relative brightness of diffuse and specular reflection can be predicted in terms of these material parameters. It has always been known that specularities are typically brighter than diffuse reflection and there are a number of image understanding techniques that segment specularities using contrast thresholding without real physical motivation for selecting such thresholds. Specularity-diffuse contrasts can be precisely predicted from our physical model as described in [20], [22]. Even if the material surface and illumination conditions are completely unknown, we have derived physical lower bounds for specularity-diffuse contrasts below which it is physically impossible for specularity-diffuse contrasts to occur. This may be useful as an added feature to image understanding techniques in discerning specular reflection on dielectric surfaces.

3 3-D Stereo Correspondence Utilizing Photometric Ratios as an Invariant

Correctly corresponding points on a smooth featureless surface utilizing intensity values between a stereo pair of images can in practice be very difficult to do for a variety of reasons having to do with the nature of video cameras and object reflectance. Influencing image grey values are F-stop, image plane-to-lens distance, angle between incident light on a pixel and the optic axis in a perspective image, and, camera gain, all of which can be at least slightly variable in an unpredictable way across a stereo pair of cameras. In addition, as was seen in the last section with respect to our research on diffuse reflection from dielectrics, such reflection is in fact viewpoint dependent.

The work of Wolff and Angelopoulou [24] (in these proceedings) shows that photometric ratios produced from diffuse reflection from different but not necessary to be known illumination conditions are reliable for accurate correspondence of object points along epipolar lines across a stereo pair of images. The methodology utilizes multiple stereo pairs of images, each stereo pair taken of exactly the same scene but under different illumination. With just two stereo pairs of images taken respectively for two different illumination conditions, a stereo pair of photometric ratio images can be produced; one for the ratio of left images, and one for the ratio of right images. We show that such photometric ratio images are invariant to changes in video camera parameters listed above. Because formula 1 above is a separable function in variables incidence angle, ψ , and emittance angle, ϕ , photometric ratios of diffuse reflection are invariant to varying viewpoint as well. We show that object points having the same photometric ratio with respect to two different illumination conditions comprise a well-defined equivalence class of physical constraints defined by local surface ori-

entation relative to illumination conditions. Stereo correspondence of equal photometric ratios is in essence the correspondence of equivalent physical constraints across a stereo pair of images without ever having to know explicitly what these physical constraints actually are.

The advantage of using photometric ratios for stereo correspondence is that it is a robust way of obtaining a dense 3-D depth map of smooth featureless surfaces, something which is normally hard to do from image feature correspondence (e.g., edge correspondence). While two different illumination conditions are required, these conditions can be arbitrary (e.g., extended light source, point light source, etc...) and never need to be known, and this technique works in full perspective views. We demonstrate in [24] experimental 3-D depth determination of a dense set of points using our stereo technique on smooth objects of known ground truth shape that are accurate to well within $\pm 1\%$ relative depth.

We have also noticed that *isoratio curves* formed by image pixels with equal photometric ratio are invariant to surface albedo and can serve as a useful photometric invariant for object recognition [24]. Isoratio curves can be used to distinguish important geometric characteristics between different objects fairly independent of diffuse reflectance properties.

Previous work on using intensity values to determine surface shape from stereo correspondence of reflectance includes the work of Grimson [5] and Smith [12]. Ikeuchi [7] pioneered a technique called "dual photometric stereo" which utilizes photometric stereo to determine surface orientation from a stereo pair of orthographic views, and then corresponds surface orientation constraints making sure to preserve consistency between surface orientation and depth.

4 3-D Shape Recovery From Reflectance and Range Data

The problem of finding 3-D shape of a smooth object from a single intensity image is a very difficult problem even when light source incident orientation and the reflectance map is known precisely [6]. Researchers have also used depth information from range finders to determine 3-D shape [2]. We [Mancini and Wolff] [9], [10] have been exploring shape recovery methodologies that combines range and reflectance data to determine local surface orientation more accurately than is possible by each data source individually. Even more, we have extended our technique to solve simultaneously for initially unknown point light source position a finite distance away, and local surface orientation.

The first step is to get initial local surface orientation estimates from least squares fitting of local quadric surfaces to range data which has a specified range error. For the case where light source incident orientation is known, the next step is to make these initial local surface orientation estimates consistent with reflectance which is assumed to be Lambertian (a good assumption for smooth dielectrics when the angle of incidence of the light source and the angle of emittance are both within the range $0^\circ - 50^\circ$, See Section 2). To do this, for each pixel we project the initial local surface orientation estimate onto the closest point on the conic section in gradient space that is consistent with image irradiance. Then we enforce surface integrability using the method developed by Frankot and Chellappa [4]. The procedure iterates between projection onto the conic section nearest point and enforcing integrability, and typically converges after about

10-20 iterations. In simulations we have found that average local surface orientation error of initial estimates were better than cut in half by our shape recovery procedure (e.g., from over 7° average error to under 2.5° average error).

For the case where incident point light source orientation is initially unknown, after initial local surface orientation estimates are generated by least squares local quadric fitting to range data, initial estimates for incident source orientation at each point are generated by a least squares fitting to a local neighborhood of reflectance data. The position of the point light source a finite distance away is then least squares triangulated by light source incident orientation rays emanating from each surface point. Then a combination of projecting light source incident orientation and local surface orientation onto nearest points of conic sections, consistent with image irradiance, together with enforcing integrability, is iterated. In simulations we were able to achieve about 5° average incident orientation error across the surface, and reduce initial local surface orientations errors about 30%.

We want to apply this to actual experimental data using a range finder and imager.

Acknowledgements

This research was supported in part by an NSF Research Initiation Award, grant IRI-9111973 and DARPA contract F30602-92-C-0191.

References

- [1] G. D. Bernard and R. Wehner. Functional similarities between polarization vision and color vision. *Vision Res.*, 17:1019-1028, 1977.
- [2] P. Besl and R. Jain. Range image understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 430-449, San Francisco, California, June 1985.
- [3] T.E. Boult and L.B. Wolff. Physically-based edge labeling. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Maui, June 1991.
- [4] R.T. Frankot and R. Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):439-451, 1988.
- [5] W.E.L. Grimson. Binocular shading and visual surface reconstruction. *Computer Vision Graphics and Image Processing*, 28(1):19-43, 1984.
- [6] B.K.P. Horn and M.J. Brooks. *Shape From Shading*. MIT Press, 1989.
- [7] K. Ikeuchi. Determining a depth map using a dual photometric stereo. *International Journal of Robotics Research*, 6(1):15-31, 1987.
- [8] J. H. Lambert. *Photometria sive de mensura de gratibus luminis, colorum et umbrae*. Augsburg, Germany: Eberhard Klett, 1760.
- [9] T.A. Mancini and L.B. Wolff. 3-d shape and source location from depth and reflectance. In *Proceedings of SPIE Optics, Illumination, and Image Sensing for Machine Vision VI*, pages 87-98, Boston, Mass., November 1991.
- [10] T.A. Mancini and L.B. Wolff. 3-d shape and source location from depth and reflectance. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 707-709, Urbana-Champaign, Illinois, June 1992.
- [11] R. Siegal and J.R. Howell. *Thermal Radiation Heat Transfer*. McGraw-Hill, 1981.
- [12] G.B. Smith. Stereo integral equation. In *Proceedings of the AAAI*, pages 689-694, 1986.
- [13] L.B. Wolff. Polarization viewer. Patent Pending October 29, 1992.
- [14] L.B. Wolff. Surface orientation from polarization images. In *Proceedings of Optics, Illumination and Image Sensing for Machine Vision II, Volume 850*, pages 110-121, Cambridge, Massachusetts, November 1987. SPIE.
- [15] L.B. Wolff. Polarization-based material classification from specular reflection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 12(11):1059-1071, November 1990.
- [16] L.B. Wolff. *Polarization Methods in Computer Vision*. PhD thesis, Columbia University, January 1991.
- [17] L.B. Wolff. A diffuse reflectance model for dielectric surfaces. In *Proceedings of the SPIE Conference on Optics, Illumination, and Image Sensing for Machine Vision VII*, Boston Massachusetts, November 1992.
- [18] L.B. Wolff. Diffuse reflection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 472-478, Urbana-Champaign Illinois, June 1992.
- [19] L.B. Wolff. *Polarization Model for Reflected Skylight*. Johns Hopkins Technical Report CS-92-31, December 1992.
- [20] L.B. Wolff. *On the Relative Brightness of Specular and Diffuse Reflection*. Johns Hopkins University Technical Report CS 92-30, October 1992 (Submitted to Optical Engineering).
- [21] L.B. Wolff. *A Diffuse Reflectance Model for Dielectric Surfaces*. Johns Hopkins Technical Report CS-92-04, April 1992 (Submitted to the Journal of the Optical Society of America).
- [22] L.B. Wolff. Diffuse and specular reflection. *Proceedings of the DARPA Image Understanding Workshop (These Proceedings)*, April 1993.
- [23] L.B. Wolff. Polarization camera technology. *Proceedings of the DARPA Image Understanding Workshop (These Proceedings)*, April 1993.
- [24] L.B. Wolff and E. Angelopoulou. *3-D Stereo Using Photometric Ratios*. Johns Hopkins University Technical Report CS 92-29, October 1992 (Also contained in these proceedings).
- [25] L.B. Wolff and T.E. Boult. Constraining object features using a polarization reflectance model. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13(7):635-657, July 1991.
- [26] L.B. Wolff and T.A. Mancini. Liquid crystal polarization camera. In *Proceedings of IEEE Workshop on Applications of Computer Vision*, pages 120-127, Palm Springs, California, December 1992.

Image Understanding Research at the Georgia Institute of Technology*

Daryl T. Lawton
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

Abstract

Our work is concerned with several areas of image understanding: perceptual organization, motion processing, model-based vision, image understanding software, and applications in domains such as terrestrial robotics, industrial and biomedical image processing. This paper is an overview of the different papers from our lab in the IU Workshop Proceedings. We first describe the design and initial prototyping of the user interface of the DARPA Image Understanding Environment (IUE) and the tools for documentation, tutorials, and publication that will facilitate the use and adoption of the IUE. We then present different motion processing algorithms. These algorithms involve generalizing earlier work for processing translational image sequences to less restricted motions; extensions to factorization methods to allow for linear features which are less dependent on precise feature-point matching; and the incorporation of models in processing dynamic images. We finally present a set of new algorithms for range-free qualitative navigation which enable mobile robots with limited recognition capabilities to form effective spatial maps for navigation and exploration.

1 Prototyping the IUE and Tools to Facilitate Its Use

The user interface of the Image Understanding Environment (IUE) is intended to provide flexible, simple, and powerful tools for exploring data, algorithms, and systems. The general principles of object oriented design used in developing the IUE object hierarchy and programming constructs have also been applied to the interface: abstraction over common operations to provide a small number of interface objects which can be freely combined by a user. The interface has been designed to have a consistent interaction with IUE objects and their

semantics, especially the abstraction in the IUE object hierarchy. Thus, the display and browsing operations express the class similarities for objects such as images, image-registered features, and spatial objects. We are hopeful that using and becoming comfortable with the interface will not involve understanding a large number of unrelated things.

An equally important part of the user interface is what we do not intend to develop. The IUE user interface must leverage extensively off existing (and emerging) interface and graphics packages and standards. The interface must be supported by ongoing and future developments in software environments and graphical user interfaces. This is critical for the long term use of the IUE because we can depend on continuous advances in these areas that we will want to take advantage of in terms of capabilities and cost.

To realize this, the interface is being developed in terms of three levels. The **Graphics Level** is the underlying "machine independent" package for display and graphic operations which tell the screen what to do. Examples would be X, GL, OpenGL, and Phigs. The **Interface Support Level** involves packages for the creation and rapid prototyping of user interfaces and related tools which are built on top of graphics level software. This also includes the tools found in the selected software development environment such as editors and debuggers. The **Image Understanding Environment User Interface (IUEUI) Level** consists of the interface objects specialized for image understanding. This includes such things as object displays, plotting displays, several types of browsers, and structures for describing the interface context. The IUEUI consists of a small set of objects which can be freely combined for very powerful results. The specifications of these objects are relatively independent of the other two levels although much of the current prototyping and design activities are directed towards understanding how to best realize the functionality of the IUEUI objects with respect to these two levels, especially for accessibility and limiting the eventual cost of the IUE for users.

The basic functional components of the IUE interface are:

- **Displays:** These deal with mapping spatial objects and images (or sets of spatial objects and images) onto two-dimensional display windows. There are

*This research is supported by the Advanced Research Projects Agency of the Department of Defense and is monitored by the U. S. Army Topographic Engineering Center under contract No. DACA76-92-C-0016

several types of displays for displaying images and image-registered features, for plotting functional relations between attributes and components of spatial objects; and for displaying surfaces.

- **Browsers:** These deal with presenting textual and symbolic information about objects. There are different types of browsers for such operations as inspecting the values in a spatial object, for performing interactive queries with respect to databases and sets of objects, and for inspecting relational graphs and networks.
- **Interface Context Descriptors:** These are for describing the state of the interface and interface objects. Examples are such things as the current color-look-up table for a given display; the current display window or browser; and links between interface objects which describe related views.
- **Command Language and Command Buffer:** Users can control their interaction with objects using an interactive command language. This also provides a complete description of the functionality of the user interface.
- **Simplified, programmable access to GUI objects:** This is intended to provide programmer access to several of the objects commonly found in Graphical User Interface Construction Kits such as knobs, sliders, text buffers, menus. These can then be used in applications and to extend the interface

We now look at these in slightly more detail and refer the reader to the respective paper in the proceedings for a more complete discussion.

1.1 Object Displays

Object Displays are for viewing and interacting with objects by mapping them onto a two-dimensional display window. This involves nearly all IUE objects: images, curves, regions, object models, surfaces, vector fields, etc. Object displays support several types of operations for controlling the mapping of an object onto a window such as the viewing transformation; mapping values through pixel-mapping functions and color look-up tables; the specification of overlay planes; transparency effects; interacting with displayed objects through selection operations and interactive function application.

There are different types of object displays:

- The **image display** is for viewing images and image-registered features.
- The **local graphics display** displays objects by mapping their values onto parameterized graphic objects such as lines and cubes. Examples are displaying vector fields and edgels.
- The **surface display** is for displaying objects that get mapped onto mesh or rendered surfaces.
- The **plot display** is for displaying functional relations between objects. Examples are one-dimensional, two-dimensional, and three-dimensional graphs; histograms, scattergrams, and views of functions and tables.

These different types are distinguished by specific methods but all inherit a large number of similar methods from the general display class. For example, overlay operations are similar for a surface display and for an image display, although they can look quite different (In one case it appears like drawing in solid colors in image registered coordinates on top of a displayed image and in the other it would be rendering the colors onto a displayed surface). Plot displays have many similarities with object displays in terms of such things as overlays and interaction methods.

1.2 Browsers

Browsers are used for interacting with text-based or symbolic descriptions of objects. They are used for actions such as queries over set of objects, determining and inspecting relationships between objects, process monitoring, and inspecting values in an object. There are two general types of browsers: **Field-Browsers** and **Graph-Browsers** of which only field browsers are currently being implemented.

Field Browsers consist of a regular array of fields. Fields can be filled with text, icons, colors, colored text, text in particular fonts. Fields can have actions associated with them when they are selected or a user changes the values in them. We distinguish between four types of field browsers which inherit from the general Field browser class:

- **Set/Database Browser:** This is presented as an array of fields. Each row of fields corresponds to selected attributes of a particular object and each column corresponds to common attributes over the set (or database) of objects. An example would be browsing the database which describes the current active objects in the IUE to find the most recently created image from some operations.
- **Object Attribute Browser:** Each row corresponds to the value of an attribute for an object. This would usually be used for inspecting a single object.
- **Hierarchical Browser:** Useful for text-based inspection of graph structures and trees. When an item is selected, the related items (along some relational dimension) are displayed in the next column.
- **Object-Registered Browser:** This contains values extracted from a spatial object, such as the intensity values in some square neighborhood of an image. Depending on the dimensionality of the object (or relationships between component objects), this can be presented as a one-dimensional array, a two-dimensional Array, or multiple two-dimensional arrays and be used to describe curves, images, image sequences, pyramids.

1.3 Command Buffer and Command Language

Users will be able to specify all interface actions through an interactive command language and be able to access all the functionality of the interface. Display operations can be performed interactively through the command

buffer. The command language will have intelligent defaults and abbreviations (such as displaying to the current window if none is specified). In addition, the commands will be usable in non-interactive code for creating scripts and general display routines.

In the actual operation of the IUE, it is not necessary that all interactions take place through this command language: some will be invoked by menus and special keys and refer to the current display context. An important part of the design of the IUE interface entails how commands (and which commands) are mapped onto menus and other interactive devices. This is especially important since the interface will support a wide community of users ranging from naive users who are interacting with hardened applications to developers. Naive users may want many interactive devices such as specialized menus while experienced users will want more powerful, abbreviated commands. Advanced users will become very adept at shortcuts that should be provided.

1.4 Interface Context Description

Contextual descriptions of the state of the interface and the status of displayed IUE objects are used for intelligent default behaviors so users needn't specify every detail of interacting with an object and can build complex displays incrementally. Many interface operations involve accessing and setting the attributes of data structures which describe the current context for such things as the current or active window; the current mapping from objects to displays (such as the viewing transformation and color look-up table); established links between windows (for specifying the relations between displays in different windows such as window to window panning and zooming); the thickness of lines in graphic overlays; and the layout of windows and browsers on a screen.

1.5 Simplified Access to GUI Objects

GUIs (Graphical User Interfaces) generally consist of several standard types of interface widgets that can be used in the interface. The IUE interface should provide routines that allow users to access these constructs and use them in their programs and the interface. The IUE should provide simplified, interactive access to the interface objects found in GUI Kits such as sliders, knobs, buttons, menus, and text input/output fields. This includes methods which enable user code to access information from specified interface objects or to prompt a user.

1.6 Interface Prototyping Activities

We are currently prototyping many different parts of the user interface to complete the functional specification and to answer basic implementation questions about choices regarding GUIs and user interface toolkits. This will help to simplify the job of the eventual integrating contractor. For reasons of rapid development, current implementation is taking place in C and C++ on Silicon Graphics machines using the GL graphics library, Motif, and the FORMS user interface toolkit. We have been able to put up the general display object and the different browsers and hope to use these as initial browsers

and displays specialized for use with the Data Exchange Format. We are exploring extensions to GNUPlot so it is compatible with the methods associated with the general display class and can provide an inexpensive plotting package. We are also evaluating OPENGGL as a possible machine independent graphics package.

1.7 Documentation and Tutorial Tools

The IUE will be supported by on-line documentation and tutorials. The tools for implementing these will also be available for enhanced communication and publication by scientists and developers who use the IUE. While there is significant activity in developing documentation and hypermedia toolkits, they remain largely machine dependent with no clear standardization. We are developing a simple documentation tool called Knowledge Weasel (KW) which is based on Lucid Emacs 19 and existing media editing tools.

Knowledge Weasel (KW) is a presentation and authoring system designed to support annotation using several different types of media. A simple analogy for KW is reading a book or attending a lecture and being able to make diverse types of comments and annotations on the material. In reality, such unrestricted annotations and comments made with respect to real books and lectures could create a significant mess (especially if made by several different people), so in developing KW we have extended this simple metaphor in several ways. The first is to provide a general format for annotations that can include several different types of media. An annotation is a common record structure wrapped around instances of different types of media such as text files, sound, drawings, postscript files, GNU-plots, code running in the GDB debugger, and others. Annotations are implemented much as a property lists in Lisp with attributes and values and are displayed as buttons with an associated region of support. When an annotation is selected it performs an operation specific to the type of annotation selected. Annotations are created using existing media editing tools for operations such as recording a sound, drawing packages, calls to other branched processes, grabbing a portion of the screen. The second extension has been to develop different types of navigation, organization and presentation tools to keep users from being overwhelmed with a great deal of possibly irrelevant information. Users can prune the set of annotations that they want to deal with and also how they are displayed. Annotations are structured to make possible intelligent processing, perhaps eventually including rule-based processing for automatic presentation and "ferretting" of information (hence the name).

We are implementing KW on Lucid Emacs 19 which is in turn based on the X window system. Lucid's implementation of Emacs Lisp provides primitives for handling display attributes such as windows, fonts, and colors. Lucid Emacs version 19 has a built-in Lisp interpreter for Emacs Lisp and this Lisp variant provides a wide variety of primitives that are useful for manipulating text, processes, and/or files. It is available via anonymous FTP on the Internet, and is also the basis of a commercial product. Knowledge Weasel is chiefly

written in Emacs Lisp but some parts, such as the part which interacts with the operating system's lock daemon (lockd), is in C and communicates via pipes with the Emacs Lisp portion of the implementation.

We have begun using an initial version of KW to develop an on-line version of the Low Level Vision course taught at Georgia Tech. We also plan to use it as part of a computer vision algorithms course where students will select a paper from the literature, implement the corresponding algorithms and use KW to develop a tutorial presentation of the paper.

1.7.1 CD-ROM version of DARPA IU Workshop Proceedings

A significant instance of technology transfer is the DARPA IU Proceedings and workshop. For the next meeting, we hope to enhance this by having the workshop proceedings available on CD-ROM, and integrated with the Data Exchange Format, a documentation and browsing tool such as Knowledge Weasel, and, possibly, the IUE itself. This will enable an extraordinary type of paper which includes data, code, additional references, animations, and extensive annotations and cross-references.

2 Dynamic Image Processing

2.1 Translational Decomposition of Flow Fields

This paper presents a set of algorithms for processing optic flow fields by approximating them as local translations of the corresponding portions of the environment. This is theoretically interesting since it dramatically simplifies the equations for inferring motion parameters from optic flow and also supplies a low level representation of image motion that might be useful for inferring motion properties from non-rigid motions. Its practical use involves its robust nature for motion constrained to an unknown plane which characterizes much of terrestrial robotics. It can also use a small number of points for inferring motion parameters from an optic flow field.

In previous work [Lawton, 1982], we developed a technique to process relative translational motion of a sensor with respect to a stationary environment or independently translating objects. This and related algorithms [Burger and Bhanu, 1989; Jain, 1983] are based on the strong geometric constraints on image motion in the case of translation - radial motion of image features from a focus of expansion determined by the intersection of the axis of translation with the imaging surface. The technique in [Lawton, 1982] was based on a measure which described the quality of feature displacements along the radial flow paths associated with a potential axis of translation. The measure was then optimized by searching over the surface of a unit sphere where each point corresponded directly to a possible direction of translation. The optimization combined the determination of the direction of translation and the corresponding image displacements into a single, mutually constraining computation. It was possible to determine the direction of translation to within a few degrees in small image areas with a few features.

We can extend the translational processing algorithm to work with more general cases of motion by applying the translational procedure to local portions of a flow field. This allows us to associate a direction of relative environmental motion with the corresponding local portion of a flow field. We call this description of image motion the **local translational decomposition (LTD)**. This is a low level representation of environmental motion which considerably simplifies the recovery of the sensor motion parameters.

Computing the LTD begins by decomposing a flow field into small overlapping neighborhoods and then approximating the motion for each neighborhood as being produced by translational motion of the corresponding portion of the environment. This is accomplished by applying a procedure which extracts the relative direction of translational motion within small image areas over a flow field. This approximates more general motion as an array of local environmental translations, and interprets local image motions as if they resulted from translational motion of the corresponding portions of the environment. This associates with local portions of a flow field a unit vector corresponding to the direction of motion relative to the sensor of the corresponding portion of the environment. Each unit vector has an associated fit-value reflecting the validity of the translational approximation.

Once the directions of motion have been established, we can then use these as constraints to determine the actual parameters of motion and to recover the structure and layout of environmental surfaces. This is broken into four different cases: motion constrained to a known plane (the normal to the plane is known); motion constrained to an unknown plane (the normal is not known); motion constrained to surfaces which are locally planar; and arbitrary motion with no assumptions.

2.2 Interactive Model Based Vehicle Tracking

While most work in motion processing has involved very minimal assumptions about objects such as rigidity, a very important area for future work is motion processing which incorporates object models. We have begun to investigate this in the restricted domain of tracking vehicles from a stationary camera in outdoor road scenes. The key idea is that motion is a critical source of information for instantiating object models and that motion processing is in turn simplified by the constraints supplied by object models.

Processing begins with a human forming a rough interpretation of a scene by interactively manipulating models of objects such as terrain surface patches, roads, gravity, and vehicles. This initial, human-directed interpretation consists of incompletely specified two dimensional drawings of expected image features and associated three dimensional object models which are also initially incompletely specified. Once an interpretation is in place, tracking algorithms then autonomously refine and extend the interpretation. For example, a human will indicate that a particular area is a road as a two-dimensional drawing. The system will then track movement along the road and fit a constraint-based description of a vehicle to this movement. As vehicles are tracked, the three-

dimensional shape of the road can be recovered. The system can determine that a vehicle has just gone off the road (or that it is behaving inconsistently with respect to the model of a vehicle) and report back to a human about unusual occurrences or behavior that cannot be accounted for.

Object models are related by constraints specifying necessary geometrical properties and relationships between objects. The use of constraints allows for flexible object instantiation. A user can indicate a vehicle and this directs perceptual processing routines to determine the corresponding local surface orientation and roads, or he can instantiate a road segment to direct the extraction and tracking of vehicles.

The work with the local translational approximation described above as been found to be useful for tracking vehicles and determining three dimensional information. Moving vehicles can often be treated as rigid objects which are translating over short periods of time. For example, as a vehicle goes around a curve, because of turning radii constraints, the axis of rotation is often far away from the vehicle itself and the vehicle motion can be treated as a sequence of small translations corresponding to tangents of the curve of motion. The local translation-based tracker determines the direction of motion of a set of extracted image points over time, and fits their motion to an estimate of the current direction of motion of the corresponding vehicle in three dimensions. The effect of this tracker can be visualized as a unit sphere with an axis corresponding to the current direction of motion. As the vehicle and the corresponding set of points move, the position of the axis changes with respect to the sphere. We expect that this processing will work well with temporal filters since there are constraints on how quickly a vehicle can change its direction of motion. Vehicle rotation is indicated by areas of the image which show differences over time, but for which no clear axis of translation can be determined. Conversely, if there is an instantiated three-dimensional road model and a rough estimate of the position of the vehicle along the road has been established, the tangent information associated with the road model can be used to initialize the search for the axis of translation. If there is an instantiated vehicle model, it restricts the features that the local translational tracker uses.

This work will be useful for applications such as telerobotic monitoring systems where low bandwidth communication is critical. The human would produce a rough scene interpretation from sensory information from a telerobots. The resulting interpretation is a model of the world that the telerobot would refine, use to control their behavior, or report back to a human. In this way, the human directs the telerobots by initializing and constraining their processing and communication between the robot and the human takes place in the context of a shared model of the world which makes possible infrequent, semantically meaningful, and very low bandwidth communication.

2.3 Shape and Motion from Linear Features

The extraction of environmental structure and motion from a sequence of two-dimensional images is a common problem in computer vision. Typically solutions to this problem are expressed in camera-centered coordinate systems where environmental geometry is specified by the depth along an image feature's ray of projection. Unfortunately, parameters computed from this camera-centered representation are dependent upon the depth to environmental features. This leads to erroneous results for objects located far from the camera.

The recently introduced *factorization method* [Tomasi and Kanade, 1990; Tomasi and Kanade, 1992; Boult and Brown, 1992] has attempted to overcome the disadvantages associated with a camera-centered representation. This method uses a world-centered coordinate system, along with an orthogonal projection assumption, in order to compute shape and motion without the intermediate calculation of depth. A matrix of image measurements is constructed by making point correspondences between image frames. The matrix is then factored into a shape matrix and a motion matrix using Singular Value Decomposition.

One problem with the factorization method is that it relies upon accurate point correspondences between image frames. This paper introduces a method of extracting shape and motion from directionally selective linear feature correspondences. This line-based algorithm is capable of reconstructing shape and motion without computing depth as an intermediate step. In addition to the orthogonality assumption, we assume that the three-dimensional direction of gravity is known relative to each image in a motion sequence.

The algorithm begins by searching for the orientation of one of the lines in the environment. This is a one dimensional search over 180° , constrained by the projection of the line on one of the image planes. Each candidate line orientation, along with the position of gravity, forms a set of quadratic equations which constrain all the other lines, as well as the rotation between image frames. An error measure is computed from the derived line orientations and used to evaluate each shape and motion configuration. Once the line orientations and parameters of rotation have been derived, the relative positions of the lines can also be computed from simple linear equations.

3 Range-Free Qualitative Navigation

Qualitative Navigation [Kuipers and Byun, 1987; Levitt and Lawton, 1990] concerns spatial learning and path planning in the absence of a single global coordinate system for describing locations and the positions of landmarks. It is based on a multi-level representation of space, which, at its most abstract level, is based on topological properties which allow a robot to describe a location using the directions of visually salient patterns (with no associated range measurements) and then navigating using the occlusions that occur among them as a basic cue to control movement through the environment. An advantage is that the robot can use landmarks for which

exact positions can not be determined. Thus, if a robot sees a building in the distance, it may not know or be able to recognize the structure as a building or determine its exact position in space but it can still incorporate this to form an effective spatial memory. This is actually quite intuitive: it is doubtful that animals navigate by detecting landmarks, determining ranges to them, and then storing everything in a single frame of reference. It also removes the effects of incremental errors due to drift.

Our work [Lawton and Levitt, 1989; Levitt and Lawton, 1990] in qualitative navigation developed while trying to produce basic navigation and recognition capabilities in an autonomous land vehicle. Initially we worked with a terrain representation based upon an *a priori* terrain grid, which describes terrain in terms of a regular square grid of features referenced with respect to a single global coordinate system. There are several problems associated with this involving difficulties with updating a terrain grid; difficulties in establishing exact three-dimensional positions of landmarks; and dealing with the limited recognition capabilities of robots.

In this paper we describe qualitative navigation algorithms which work completely at the topological level, dealing with landmarks for which there are no range estimates. In addition, we introduce several distinctions for qualitative navigation algorithms. One type of distinction concerns landmarks. We consider two basic types: **distinct landmarks** which can always be recognized as the same from wherever they are seen and **non-distinct landmarks** which may not be recognized as being the same when seen from different points of view. We assume that once landmarks are seen, they can be tracked over time until they disappear. The other distinction involves whether or not the navigation algorithms use a compass to yield a fixed direction. We also distinguish two different types of compass. The direction associated with a **local compass** can change from place to place, but at a given place, it will always point in the same direction. An example is a compass which is effected by fixed magnetic influences at different locations. The local compass can also be a very strong landmark which is visible from a wide set of views. A **global compass** will always point in the same direction regardless of where the robot is located. We can express these distinctions as a table corresponding to the different types of topological navigation algorithms we have developed:

Topological Qualitative Navigation Algorithms

	Compass	No Compass
distinct landmarks	Very Good	Good
non-distinct landmarks	Good	Difficult!

For example, qualitative navigation without a compass and in a world of identical, non-distinct landmarks is very difficult and depends critically on matching viewframes based exclusively upon the angular orientations of landmarks. More practical algorithms are those which are based upon the use of a local compass and a limited number of distinct landmarks. This corresponds

to a freely navigating robot which can build maps and navigate using landmarks which are based on simple visual features, such as colored regions and edges aligned with gravity.

References

- [Boult and Brown, 1992] Terrance E. Boult and Lisa Gottesfeld Brown. Motion segmentation using singular value decomposition. In *Proceedings of the Image Understanding Workshop*, pages 495-506, 1992. San Diego, CA.
- [Burger and Bhanu, 1989] W. Burger and Bir Bhanu. On computing a 'fuzzy' focus of expansion for autonomous navigation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 563-568, 1989.
- [Jain, 1983] R. Jain. Direct computation of the focus of expansion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:58-64, 1983.
- [Kuipers and Byun, 1987] B.J. Kuipers and Y.T. Byun. A qualitative approach to robot exploration and map-learning. In *Proceedings of the Workshop on Spatial Reasoning and Multi-Sensor Fusion*, 1987. Morgan Kaufman.
- [Lawton and Levitt, 1989] D.T. Lawton and T.S. Levitt. Knowledge based vision for terrestrial robots. In *Proceedings of the DARPA Image Understanding Workshop*, 1989. Palo Alto, CA.
- [Lawton, 1982] Daryl T. Lawton. Processing translational motion sequences. *Computer Vision, Graphics, and Image Processing*, 22:116-144, 1982.
- [Levitt and Lawton, 1990] T.S. Levitt and D. T. Lawton. Qualitative navigation for mobile robots. *Artificial Intelligence*, 44:305 - 360, 1990.
- [Tomasi and Kanade, 1990] Carlo Tomasi and Takeo Kanade. Shape and motion without depth. In *Proceedings of the Image Understanding Workshop*, pages 258-270, 1990. Pittsburgh, PA.
- [Tomasi and Kanade, 1992] Carlo Tomasi and Takeo Kanade. The factorization method for the recovery of shape and motion from image streams. In *Proceedings of the Image Understanding Workshop*, pages 459-472, 1992. San Diego, CA.

Image Understanding Research at University of Washington

Robert M. Haralick
Intelligent Systems Lab
Dept. of EE, FT-10
University of Washington, Seattle, WA

Abstract

Recent progress in image understanding research at the University of Washington is described in this paper. The main focus of the research has to do with performance characterization of computer vision algorithms. We provide an overview of our approach to performance characterization and discuss ongoing theoretical and experimental work.

1 Introduction

Our current focus in Image Understanding research at the University of Washington is on performance characterization of computer vision algorithms. Our present research objective is to develop the methodology for evaluating the performance of image understanding algorithms and systems. In the first section of this paper we summarize the general approach we use for performance characterization. Subsequent sections discuss accomplished and ongoing work in performance characterization

at University of Washington.

2 Performance Characterization

Image Understanding systems employ different algorithms applied in sequence. Determination of the performance of the complete IU algorithm is possible if the performance of each of the sub-algorithm constituents is given. An algorithm employed at any stage in the image analysis sequence employs a representation for the data with which it works. In our approach, we address questions such as: what kinds of conditions exceed the limits of the representation? When is reality not covered by the representation? What conditions make numerical computations, that the algorithm performs, to be unstable? Finally, since the algorithm works with noisy data, data which has been perturbed from its ideal form, the results of the algorithm will be perturbed from their ideal form too. To what degree will a perturbation

of the input data affect the accuracy of the output, in a qualitative sense and in a quantitative sense?

The methodology involves both black-box and white-box perspectives. In the black-box perspective, the problem is one of determining what the requirements of a computer vision task are, and performing empirical evaluations to verify whether these requirements are met. In the white-box perspective, the algorithm is examined from the inside out. Under a given set of well-defined assumptions, does the algorithm provide guaranteed answers? This is done by performing a theoretical evaluation of the algorithm. The consistency of these assumptions used in the theoretical analysis with the reality that the algorithm is supposed to handle is established by a experimental reality-assumption validation test.

What does performance characterization mean for an algorithm which might be used in a IU system? Each algorithm is designed to accomplish a specific task. If the input data is perfect and has no noise and no random variation, the output produced by the algorithm should also be perfect. Otherwise, there is something wrong with the algorithm. So, measuring how well an algorithm does on perfect input data is not interesting. Performance characterization has to do with establishing the correspondence of the random variations and imperfections which the algorithm produces on the output data caused by the random variations and the imperfections on the input data. This means that to do performance characterization,

we must first specify a model for the ideal world in which only perfect data exist. Then we must give a random perturbation model which specifies how the imperfect perturbed data arises from the perfect data. Finally, for the last algorithm in a vision sequence we need a criterion function which quantitatively measures the difference between the ideal output arising from the perfect ideal input and the calculated output arising from the corresponding randomly perturbed input.

The difficulty in performance evaluation is in deciding what the appropriate random perturbation model must be for each input or output a vision algorithm component may have. Sometimes the algorithm component may change data structures from input to output. This means that the random perturbation model must be different from input to output. And of course, the choice of the random perturbation model for a vision algorithm component output must be suitable for the input to the subsequent vision algorithm component. Very quickly one finds that the classical Gaussian models are not appropriate.

3 Current and Ongoing Work

Recent work, [1], was focussed on the white-box perspective. We were interested in setting up random perturbation models for typical computer vision algorithms and relating the model parameters to performance measures of algorithms. In the past year, we theoretically analyzed an ex-

ample vision an example vision algorithm sequence that involves edge finding, edge linking, and line/arc fitting. By starting with an appropriate noise model for the input data we derived random perturbation models for the output data at each stage of our example sequence. These random perturbation models are useful for performing model based theoretical comparisons of the performance of vision algorithms. Parameters of these random perturbation models are related to measures of error such as the probability of misdetection of feature units, probability of false alarm, and the probability of incorrect grouping. In Ramesh and Haralick [1], we described a theoretical model by which pixel noise can be successively propagated through an edge labeling algorithm, an edge linking algorithm and a boundary gap filling algorithm. Assuming an edge idealization of a linear ramp edge and i.i.d Gaussian random perturbations on pixel gray values it was shown how one could model the breakage of a true line segment as a renewal process with alternating segment and gap intervals. It was also shown that if one ignores the dependencies between adjacent gradient estimates then the segment and gap interval lengths are exponentially distributed with parameters that are related to the true edge gradient, the neighborhood operator size and the gradient threshold employed. It was also shown how the output after a gap filling operation could also be modeled as an alternating renewal process and we derived the expressions for the probability distributions of the lengths of segment and gap

intervals obtained after the filling operation.

In reality, there is an overlap between the edge detector neighborhoods centered around pixels. Hence there is some dependence between gradient estimates obtained for neighboring windows. In addition, if one assumes that the noise at each pixel is locally dependent then the correlation in the noise would introduce correlation in the gradient estimates. In addition, the analysis in [1] did not include positional errors. These positional errors are of significance if one wishes to analyze higher-level matching algorithms. Hence, we extended the results presented in [1] to handle dependencies between gradient estimates for neighboring edge pixels. Under the assumption that the gradient across the edge is constant along the entire model line segment, we illustrated how the dependencies between neighboring pixels can be captured by modeling the sequence of labeled edge and non-edge pixels along the true model line as a binary Markov Chain of a particular order. The transition probabilities for the Markov chain are shown to be related to the true edge gradient, the edge operator width, the noise variance, and the edge operator threshold.

In other work, [2], we focussed on performing theoretical model-based comparison of gradient based edge finding schemes and mathematical morphology based edge finding schemes. The performance analysis was done by assuming an ideal edge model and a noise model and by deriving expressions for probability of false alarm and probability of misdetection of edge

pixels. Under the Gaussian noise model assumption, the theory indicated that the morphological edge detector is superior to conventional gradient based edge detectors, that label edges based on gradient magnitude, when a size 3 by 3 window was used. We performed experiments to validate our theoretical results and the empirical plots indicated that the morphological edge operator was also superior when a 5 by 5 window is used. However, the theoretical plots did not confirm this because the theory provided only an upperbound. In [2] we also included comparisons of results obtained for real images. A simple analysis of hysteresis linking was also done in this paper and it was shown that hysteresis linking improves the performance of the edge operators.

Our recent work involved the design and implementation of an experimental protocol to compare the accuracy of the edge locations obtained for the two operators. We generated synthetic images containing ramp edges of various orientations and additive noise of varying degree. We defined the edge pixel location error as the distance along the gradient direction from the true edge pixel to the nearest labeled edge pixel (if one exists) in the edge detector output. We applied the morphological edge operator and the gradient based operator on these images and we are in the process of evaluating the results.

We are also in the process of evaluating line finding schemes. We have devised an experimental protocol for evaluating line finders and are conducting the experiments now. Specifically, we have gen-

erated synthetic image data consisting of ramp edges with varying orientations and additive Gaussian noise of different levels. We are in the process of estimating the random perturbation model parameters at each stage of the line detection schemes employed. The results from these experiments will be used to validate the theory discussed in [1]. We plan to perform similar evaluation using RADIUS imagery.

In the black-box mode of performance characterization, we have defined the meaning of an experimental protocol. We have set up an experimental protocol for evaluating the performance of an algorithm which computes the exterior orientation given a set of 3D model points with its corresponding 2D perspective projections [3]. The exterior orientation algorithm computes the rotation and translation by which the model reference frame can be transformed into the camera reference frame. The experimental protocol in [3] illustrated how ideal data could be randomly generated and how this ideal data was randomly perturbed. The random perturbations employed included both small perturbations, that affected most of the generated perspective projection data points, and large perturbations that affected a small part of the generated data points. Experiments were conducted to compare the standard iterative equally weighted least-squares against the iterative reweighted least-squares technique.

Planned future work includes evaluation of algorithms for circle finding, ellipse finding, and rectangle finding, particularly as these algorithms are employed in the un-

derstanding of aerial images. In this application rectangles correspond to roof tops and circles and ellipses correspond to spherical holding tanks or circular chimneys.

References

- [1] V.Ramesh, R.M.Haralick, "*Random Perturbation Models and Performance Characterization in Computer Vision*," Proceedings of the CVPR conference, June 1992.
- [2] V.Ramesh, R.M.Haralick, "*Performance evaluation of edge operators*," Special Session on Performance Evaluation of Modern Edge operators, Orlando Machine Vision and Robotics Conference, 20-24 April 1992.
- [3] R.M.Haralick, V. Ramesh, and K. Thornton, "Performance Characterization Protocol in Computer Vision," Unpublished Manuscript.
- [4] J.S.Lee, R.M.Haralick, and L.G.Shapiro, "Morphologic Edge Detection," *IEEE Journal of Robotics and Automation*, Vol. RA-3, No.2, April 1987.

Model-based Recognition of Objects in Complex Scenes

Thomas O. Binford, Tod S. Levitt *

Robotics Laboratory, Computer Science Department,
Stanford University, Stanford, CA 94305, U.S.A.

Abstract

A goal of this research is effective recognition of complex objects in realistic, operational scenes with moderate complexity, using general methods with a rigorous scientific basis. This research is intended to contribute to ATR, mapping and site monitoring. Major research issues are: 1) robust recognition despite image complexity; 2) exploitation of multi-spectral, multi-sensor data; 3) low-complexity algorithms; and 4) automation of development of recognition programs.

The technologies of this research that contribute to resolving these major issues are: a) quasi-invariants; b) structured Bayesian inference; c) segmentation and measurement, and d) shape representation.

From the point of view of computational complexity, the most important problem in recognition is figure-ground discrimination. In the last IU Workshop, effective figure-ground discrimination was demonstrated based on quasi-invariants derived for Generalized Cylinder parts (GC). The shape of 3d objects was inferred from monocular images [Sato and Binford 92a,92b].

New theoretical results have been achieved in quasi-invariants, including: 1) a new mathematical definition of quasi-invariants; 2) derivation and proof of two new strong quasi-invariants for four coplanar points; 3) results about the taxonomy of quasi-invariants.

Recognition was demonstrated for an aircraft at San Francisco airport and canisters in a complex crash image, based on quasi-invariants. Progress has been made toward the goal of recognition by Bayesian networks that are generated automatically from object models. A

new object and constraint system, *Classics*, was implemented to facilitate the geometric reasoning necessary to generate Bayesian networks automatically.

Extended edges for a variety of scenes were generated using a preliminary local linking of edgels from a new Wang-Binford operator. Extensive performance evaluation was done to build a statistical model for the new Wang-Binford operator to incorporate in Bayesian networks. Image measurements of position and orientation are an order of magnitude more accurate than those in ACRONYM, Stanford's system from 1980, making possible an order of magnitude more accurate estimates of part dimensions and stereo measurement. Effective measurement appears possible to a few percent for surfaces with images extending only 5x10 pixels.

I: Introduction

The goal of this research is to develop effective methods to contribute to ATR, cartography and surveillance. This requires effective recognition, interpretation and measurement of complex objects in realistic, operational scenes with moderate complexity, using general methods with a rigorous scientific basis. Considerable progress has been achieved.

Major research issues are: 1) robust recognition despite image complexity; 2) exploitation of multi-spectral, multi-sensor fusion; 3) development of low-complexity algorithms; and 4) automation of development of recognition programs.

The technologies of this research that contribute to implementing mechanisms that solve those problems are: a) quasi-invariants; b) Bayesian inference; c) segmentation and measurement; and d) shape representation. This research takes an integrated system approach to building component technology and the Successor system.

Robust recognition depends on accurate measurement of object dimensions, based on both accurate measure-

*This research was supported in part by a contract from the Air Force, F30602-92-C-0105 through RADC from DARPA SISTO, "Model-based Recognition of Objects in Complex Scenes: Spatial Organization and Hypothesis Generation" and supported in part by a contract with NASA-AMES Research Laboratory NCC 2-574: "Nap of the Earth Flight by Helicopter".

ment of image dimensions and effective interpretation of image features as object surfaces. Effective segmentation based on minimum variance methods enables accurate measurements of image dimensions. A benefit of this research is effective, robust recognition with fewer pixels on target than possible with other methods. Quasi-invariants enable generation of hypotheses assigning image features to object surfaces, and allow estimating shape parameters in space. Quasi-invariants also enable new stereo correspondence.

Bayesian inference networks that incorporate quasi-invariants [Binford 87a] with physical constraints were motivated multi-sensor and information fusion.

Recognition is fundamentally limited by computational complexity. Complexity of brute-force, view-sensitive methods is enormous (e.g. aspect graphs). View-sensitive methods for pose estimation match all combinations of image features with all views of all objects. The dominant contribution to computational complexity is matching all combinations of image features. For typical aerial imagery, scene complexity dominates over the number of views of objects, i.e. the number of combinations of m features from all image features is much larger than the number of views of all objects. Our research provides quasi-invariant mechanisms to reduce this combinatorial complexity enormously by figure-ground discrimination in clutter to match only groups of features belonging to a single object to avoid matching all combinations of image features. From the point of view of computational complexity, figure-ground discrimination (grouping) is a central problem in recognition.

A smaller yet significant part of the computational complexity in recognition by view-sensitive methods is matching the number of views of objects. Invariants, where available, are view-invariant; they avoid the computational complexity of view-sensitive methods and enable indexing for object hypotheses. Quasi-invariants extend invariants greatly, making these view-insensitive methods much more widely applicable. Avoiding matching all views of all objects is the problem of hypothesis generation and indexing for similar objects.

From another standpoint, the inability of view-sensitive methods and pose estimation to accommodate variability of natural scenes is also very important. Quasi-invariants allow considerable variation within object class and viewing conditions.

Considerable progress has been made toward automating the building of recognition programs from object models.

II: Approach

The system achieves recognition by a hypothesis generation and verification paradigm implemented in Bayesian decision networks. There are four modules: 1) figure/ground discrimination of generalized cylinder primitive parts (GC) from background based on quasi-invariant relation among extended curves along image discontinuities; 2) estimation of 3D shape of GC primitive parts from 2D image data based on quasi-invariants;

3) generation of object hypotheses by indexing based on estimation of shape of compound objects from shapes of GC parts; and 4) verification or refutation of object hypotheses by Bayesian networks.

At the last IU Workshop, results were demonstrated for figure-ground discrimination for a variety of images. The methods rely on rigorous quasi-invariant relations among curves on cross sections (parallels) and meridians of GCs. The estimation of shape of cross sections was demonstrated also. Those results have been extended in two ways: 1) A new algorithm for determining relations among curves has been designed and implemented. It is used for several recognition examples and cuts the computational complexity dramatically. An extended algorithm for figure-ground discrimination has been designed but not implemented; it extends the applicable class of GCs greatly. 2) New quasi-invariants and improved probabilistic characterization of quasi-invariants extends estimation of 3d shape from 2d image data.

Substantial progress has been made toward automating the building of Bayesian decision networks from object models. A new object and constraint system, Classics, was implemented to facilitate the geometric reasoning necessary to generate the Bayesian networks automatically. Examples of recognition presented at the last IU Workshop will soon be fully automated, based on Classics models.

Extended edges for a variety of scenes were generated using a preliminary local linking of edgels from a new Wang-Binford operator. Extensive performance evaluation was performed to build a statistical model for the new Wang-Binford operator to enable its use in Bayesian networks. Linking edgels into extended edges has proved very difficult in IU. Complexity of linking is exponential in the error of position and orientation of edgel measurements. There is a high priority and payoff in accurate measurement that is typically overlooked.

Improved estimation improves recognition by making more reliable discrimination of object parts from clutter and to make more accurate estimates of part dimensions. These dimension measurements are typically an order of magnitude more accurate than in typical IU systems, typically 1% error in for an image region 10 pixels. It is quite feasible to work with regions 5x10 pixels with image measurements to a few percent. With order of magnitude better resolution now, routine counting of types of aircraft at airports seems at hand. Stereo reliability and accuracy in height measurement are aided in the same way.

III.1.a: Quasi-Invariants: Theory

Quasi-invariant observables are also called semi-invariants or local invariants. Quasi-invariant observables are locally invariant about some observation. The explanation follows. an observation is a measurement made from some viewpoint. An observable is a measurement repeatable by different observers with coordinate frames that differ by rotation and translation. Observables are invariant under rotation and translation, i.e.

isometries.

Invariant observables under perspective projection in imaging are observables that have a constant value from all projections. E.g. for four colinear points, the cross ratio is invariant. Quasi-invariant observables under perspective projection are observables that are constant locally about some projection, i.e. the gradient of the observable with respect to projection parameters vanishes at that projection. E.g. for three colinear points, the ratio of intervals is quasi-invariant about projection for a distant line parallel to the image plane.

Quasi-invariants were introduced by Binford in the late 1960s to extend algebraic invariants and were used numerous times since. The definition given in [Binford 93] is a refinement of the mathematical definition used in the past. Over the twenty years of their use, a large number of useful quasi-invariants were invented by the author.

Several new invariants were derived. The most powerful are 2 strong quasi-invariants for 4 points in a plane. These results extend invariants: there are known to be two invariants for five points in the plane. There are known to be no invariants for four points and three points in the plane. There were known to be two quasi-invariants for three points in the plane [Binford 87a]. For four points in the plane there are four quasi-invariants. The two new quasi-invariants are strong. For strong quasi-invariants all second derivatives with respect to viewpoint parameters vanish [Binford 93]. Strong quasi-invariants are nearly invariant. These quasi-invariants are expected to be extremely valuable.

A taxonomy, a classification scheme, for quasi-invariants was developed. The taxonomy includes: invariants, generic observables, strong quasi-invariant observables, and quasi-invariant observables.

We are investigating not only the local behavior of quasi-invariants but their global behavior as well. If quasi-invariants were stable only in a small region around the observation, they would have limited value. It turns out that quasi-invariants investigated to date are "stable in large measure", e.g. they are constant to 30% over $\frac{2}{3}$ of the viewing sphere. Strong quasi-invariants are much more nearly constant. The global analysis of the colinear ratio, a strong quasi-invariant, showed that the standard deviation is 1% over a typical human limit, that the projected colinear ratio is invariant to within 30% over almost the full range of viewing for an object 2 miles long in imagery from aircraft flights, i.e. for extremely large objects. The projected colinear ratio is much more nearly invariant for smaller objects.

These results are very favorable. There is reason to believe that these results are widely true for quasi-invariants and strong quasi-invariants. Systematics of global analysis of quasi-invariants are under investigation now [Binford 92].

III.2.b: Quasi-Invariants: Exploitation

[Mundy et al 92] demonstrate recognition of buildings using invariants for five points in a plane. A rectan-

gular building has only 4 points, but there might be small structures on the roof. An L-shaped building has 6 points in a plane; a U-shaped building has 8 points in a plane. In these examples, there are typically 10,000 points or lines in a large image. The number of combinations of 5 points is $\binom{10,000}{5} \approx 10^{18}$ calculations of 2 invariants and 2d index operations. That number is prohibitive. Simple grouping was used to reduce the number of combinations, corresponding to curvilinearity.

By contrast, for 10,000 objects, there might be about 10^6 total views. That is a very large number but miniscule in comparison with the number of combinations of image features [Binford 93]. Conclusion: image complexity dominates over the number of views of objects. From the computational point of view, figure-ground discrimination is the most important operation in computer vision.

Quasi-invariants based on generalized cylinders provide the basis for figure-ground discrimination. Quasi-invariants also enable the estimation of shape parameters for parts and objects, enabling indexing to generate object hypotheses to reduce the complexity of matching all views of objects.

An obvious way to recognize objects is to make measurements of object dimensions in space and compare them with dimensions in tables of objects. These measurements of object dimensions are Euclidean invariants, i.e. invariant under rotations and translations. There is obviously no quasi-invariant for length, a usual Euclidean invariant, but there are quasi-invariants for ratios of lengths. For many Euclidean invariants, such quasi-invariants have been found. The investigators believe that it will be possible to develop a systematic method to generate quasi-invariants, just as has been developed for some types of algebraic invariants.

One big advantage of quasi-invariants is that there are many of them and they appear to be widely available, i.e. quasi-invariants are there when needed, for most objects in most situations. There are few invariants.

Another advantage of quasi-invariants is that exploitation is intuitively clear once the paradigm switch has been made. Recognition is done by interpreting quasi-invariants as approximate measurements of objects in space. Quasi-invariants determine figure-ground discrimination, i.e. generating hypotheses of object parts. Quasi-invariants determine hypotheses about proportions of object parts and objects, generating hypotheses of object shape that enable indexing.

The paradigm shift is from brute force matching feature sets from models to all combinations of feature sets in the image domain, to generating object hypotheses, generating 3 space descriptions of objects from images, indexing to generate hypotheses of matching objects, and detailed verification.

As a simple example, invariants are not possible for any plane figure with three or four points, e.g. a triangular cross section or quadrilateral. There are two quasi-invariants for three points in a plane; there are two additional strong quasi-invariants for four points in a plane. Thus, approximate measurement of ratio of dimensions of a triangular face and recognition based on

these measurements are possible for monocular images, and very good approximate measurements for quadrilateral faces are possible. For five points or more in a plane, invariants are possible also. These results are very powerful. They are complete in a mathematical sense in that any polygon can be recognized approximately with these methods. Quasi-invariants give shape measures that can be used for indexing. New results are expected for non-polygonal faces to extend completeness results.

A major use of quasi-invariants has been in stereo vision [Arnold and Binford 80, Baker and Binford 81]. Projected angles in two stereo views of an edge in space are nearly equal, i.e. there is a narrow distribution of the difference of angles in two views, with 1 degree full-width half-maximum for human stereo at 1 meter. There is much more to be done in stereo using quasi-invariants.

Generalized cylinders are used to describe a very large class of objects. Generalized cylinders (GCs) express powerful quasi-invariants for grouping image features that define a mechanism for discrimination of GC parts, figure-ground discrimination. Generalized cylinder primitives are defined by cross section and sweeping rule. Quasi-invariants permit estimates of shapes of GC parts and objects formed of GC parts that enable generation of hypotheses that are verified by globally coherent interpretation by a Bayes net for evidential inference.

An important part of this paradigm of use of quasi-invariants follows. Quasi-invariants are inexact; a quantitative probabilistic interpretation enables efficient use of quasi-invariants in recognition in a paradigm of hypothesis generation and verification. The statistical analysis is for hypothesis generation, not accuracy of the final match. This is an important distinction. For a majority of cases, quasi-invariant observables are approximations to corresponding body measurements in space; those quasi-invariants generate good approximate hypotheses. Distributions of deviations are known or can be derived for use in evidential inference. For a minority of cases, quasi-invariant observables give bad estimates of corresponding body measurements; those quasi-invariants generate false hypotheses that are rejected by a verification phase, in most cases at low cost. This hypothesis generation and verification mechanism depends on making some accurate measurements and some accurate hypotheses; the mechanism tolerates local errors. Computational complexity can be low.

III.2: Bayesian Networks and SUCCESSOR System

The objective of this part of the effort is provide a comprehensive way of integrating available information, performing sensor fusion. A further objective is to solve the associated software problem by automating the building of Bayesian decision networks from object models.

III.2.a: SUCCESSOR System

We are working toward automated, model based 3-D interpretation of imagery. The algorithms are intended for multi-sensor fusion; here algorithms are tested with

monocular gray scale images, a difficult problem. The algorithms described in this section begin with linked edges, group them into GC surfaces, and ultimately establish a 3-D object interpretation of the surfaces. There are three primary components in this section of our interpretation system: the Bayesian network interpretation kernel, the Volume-Surface-Curve-Point (VSCP) modelling system, and Classics, a highly-typed object and constraint system.

The Bayesian network is a graph structure of object model hypotheses and conditional probabilities between the hypotheses. Geometric and physical models determine geometric constraints with detailed probability models for measurements and estimation algorithms. Bayesian networks are at the core of our interpretation approach; they have been demonstrated successfully in the 3-D interpretation of an aircraft in optical imagery from San Francisco airport and in 3-D interpretation of a plastic elbow, a plumbing part. Work will be completed soon that will generate Bayesian inference networks automatically from the VSCP modelling system.

The VSCP graph (Volume-Surface-Curve-Point) represents the topology of neighbor relations among the topological types of objects. It is an intermediate level of modeling that will be derived from object models. The goal of the VSCP modelling system is to define geometric information to permit automated computer reasoning and interpretation, and do it in a way that is concise and usable. To achieve this goal we have built our models up using mathematically sound definitions. In addition, we have developed a special constraint system, Classics, to permit the necessary level of symbolic expression.

The Classics system is a highly typed constraint system. It supports the definition of object classes in terms of other classes using constraints.

We have achieved 3-D interpretation of an aircraft in an airport scene and a plastic elbow, a plumbing part. We used Bayesian networks to guide the recognition, accumulate evidence and provide a measure of the most likely interpretation of the data. These results are shown in the figures below. These results demonstrate the viability of aspects of our approach: a) generating object hypotheses from quasi-invariants; b) accurate probability models; and c) using Bayesian networks to aggregate evidence both in support of correct hypotheses and in refutation of incorrect hypotheses.

Interpretation proceeds by: 1) generating part hypotheses by finding corresponding curves from quasi-invariants that make generalized cylinder part hypotheses; 2) estimating part measurements from quasi-invariants; 3) generating object hypotheses from part hypotheses and indexing based on object shape from quasi-invariants to determine object model hypotheses; 4) verification in Bayesian networks. All of these steps 1-4 use generic models that utilize information available at that stage. None of these steps is specific to object models; all steps accommodate great variation in the generic models. This approach avoids the combinatorics of attempting to match every combination of image features with every view of every known object model. In addition, at every level of interpretation, the model is used to pre-

dict and refine edge and image information to aggregate more data. Strong data give rise to hypothesized models, which are used to predict and include other data.

Probabilistic relationships between the models and the data are expressed by the Bayesian Network. Probabilistic relationships lead to a mathematically sound algorithm and avoid the many ad-hoc decisions common to recognition algorithms. Generic methods for creating 3-D interpretations from 2-D data avoid the computational problem of too many false hypotheses.

III.2.b: Classics

We have completed an implementation of Classics, a highly typed object system and constraint system. Classics can be thought of as a combination of an object-oriented data base and a symbolic declarative programming language. The goal of Classics is to support the creation of a geometric modelling system with sufficient power and symbolic representation to perform true model-based recognition. Classics is written in Common LISP on top of the Common LISP Object System (CLOS) as implemented by Portable Common Loops (PCL). Classics is unique in that it not only defines an object class hierarchy, but does it in a way that solution algorithms can be derived automatically from the constraints that define classes.

We are building the Volume-Surface-Curve-Point (VSCP) object modelling system using Classics. From the VSCP the algorithm will automatically derive the Bayesian network in real time and control the interpretation. The goal is to provide automatic image interpretation from object models. At this intermediate step, the system builds a Bayesian network from VSCP models. The user creates high level models of specific manufactured parts. All of the geometric relationships and observable features of the part will be extracted automatically. The system will automatically integrate the part into the probabilistic characterization of the low level models.

III.3 Segmentation

What seems like an incomprehensible variety of imaging problems breaks down into a small set of segmentation components based on careful analysis of physics and differential geometry of observation [Binford 87b]. There is much in common between segmenting depth data, segmenting intensity data, and segmenting SAR and IR data. Multi-sensor segmentation is achievable. Complete segmentation is feasible in an interesting sense, not in the sense of perfect or faithful segmentation, for which information might not be available. Instead, a complete segmentation is possible in what physics and differential geometry make possible a small enumeration of local image configurations.

Extended edges for a variety of scenes were generated using a preliminary local linking of edgels from a new Wang-Binford operator. Extensive performance evaluation was done for the new Wang-Binford operator. Im-

proved estimation accuracy results in improved and more reliable discrimination of object parts from clutter and more accurate estimates of part dimensions.

III.3.a: Local Discontinuities

Binford and Wang [Wang and Binford 93] have developed a step edge estimation operator, starting from a modified Canny operator. They have improved sensitivity by a factor of 4, i.e. decreased the threshold for gradient magnitude by a factor of 4. The only parameter used is a measured parameter, i.e. sensor noise variance. There are no free parameters, no tweaking. Those parameters are measured from the sensor or measured from image content. Analysis has eliminated biases from orientation and position, biases that are so strong that several researchers do not use orientation from the Canny operator. With the Wang-Binford operator, orientation is accurate to a few degrees over a wide range of conditions.

A range image, SAR or IR image, or optical intensity image is a compound mathematical surface. Differential geometry applied to the physics of image formation allows us to classify the local classes of image surface. A small image patch corresponds to: 1) a single continuous image surface; 2) a pair of image surfaces with an edge discontinuity at their intersection; a single image surface with a spot discontinuity; 3) three or more image surfaces with curve discontinuities meeting at a vertex discontinuity; 4) more complex configurations that may not be discriminable with fixed image resolution.

The conclusion is that robust segmentation for a variety of imagery can be achieved with a small number of local segmentation components. The objective of this part of the effort is: a) to design and implement the complete set of local segmentation components; b) to design and implement the linking method that constructs quasi-local and global segmentations from local segmentations. Segmentation corresponds to generating hypotheses about the image surface. Interpretation corresponds to generating hypotheses about surfaces in space and objects from image surfaces. An important part of this work is building an accurate statistical model of behavior of segmentation that is valuable in the Bayesian network for combining evidence, especially in multi-sensor problems.

Edge discontinuity detection extracts boundaries of image surfaces and discontinuities of orientation of boundaries, etc. It has played an important role in the early process of a vision system, and has a direct impact on performance of subsequent processes. When we look at the output of typical edge segmentation algorithms on simple images, they look adequate, superficially. When we take a close look at their performance in complex images we find important faults that can be overcome. For example, the Canny operator, which depends on the gradient of the image values, is sensitive to shading. This causes false edgels where there is no discontinuity, on curved surfaces like the fuselage of an aircraft. The effect of gradients also causes large biases in the estimate

of orientation and position of edgels. These biases contribute to inaccuracy of measurement that induces an exponential complexity to linking and to recognition. The reason for the biases is that the Canny operator uses an inaccurate and incomplete model of the local image surface. The Canny operator is designed for a step edge between flat image surfaces. Such edges are a minority. On any other edges, the operator has biases from the inappropriate model.

The Wang-Binford algorithms are two directional operators designed to minimize inaccuracy including biases in edgel orientation and position estimation from a raw image. The second of the two algorithms will be described here. It uses 2-dimensional fitting, instead of 1-dimensional fitting, that is less sensitive to factors which are not included in a step edge model. The operator also uses the logarithm of the gradient magnitude to map a nonlinear fitting problem to a linear problem, thus reducing the computational complexity greatly.

This operator is effective for step edges in intensity images. It is not complete in the sense that the operator is not effective for many image features, e.g. lines or spots. The author has promoted development of a complete set of operators [Herskovits and Binford 69].

A Gaussian derivative is convolved with the intensity image to estimate gradient components in i and j directions. An estimate is made of the local surface curvature near local maxima of the log of the gradient magnitude. The gradient magnitude along the transverse direction of a step edge is a Gaussian; its logarithm is parabolic. A linear 2-dimensional quadratic surface is fit to every maximum with a 3 by 3 support to get the position, orientation, and contrast information of the edgel.

Estimate of improvement of the Wang-Binford algorithm is underway by building statistical models of this algorithm and others, i.e. the Roberts-Cross operator, the Canny Operator, etc.

Conclusion

Quasi-invariants from GC shape representation provide a basis for figure-ground discrimination of GC parts hypotheses. Quasi-invariants also enable estimation of 3d shape of parts and objects that enables view-insensitive hypothesis generation and indexing. These two mechanisms reduce drastically the first and second most computationally complex operations in recognition in moderately complex imagery. They also enable recognition with targets with a high degree of variability.

Classics implement an object system with a strong mathematical type heirarchy. The shape representation implemented in Classics enables automation of target-specific recognition from VSCP object models.

Improvements in segmentation demonstrating extended edges are important to robust recognition and measurement.

We plan to demonstrate image interpretation of more complex object models, including multiple parts with occlusion, within the next quarter. The recognition will be completely automatic and model-driven.

IV: References

- [Binford and Brooks 79] Binford, T.O., Brooks, R.A.; "Geometric Reasoning in ACRONYM"; Proc Image Understanding Workshop, Palo Alto, Calif, 1979.
- [Arnold and Binford 80] Arnold, R.D., Binford, T.O.; "Geometric Constraints in Stereo Vision"; Proc SPIE Meeting, San Diego, California, July 1980.
- [Binford 81] T.O.Binford; "Inferring Surfaces from Images"; Artificial Intelligence Journal August, 1981.
- [Binford 82] T.O.Binford; "Survey of Model-Based Image Analysis Systems"; Int J. of Robotics Research, v 1, p 18, 1982.
- [Binford 87a] T.O. Binford, T.S. Levitt, W.B. Mann; "Model-Based Bayesian Inference in Computer Vision"; proceedings Uncertainty in AI Workshop at AAAI 87; reprinted in Uncertainty in Artificial Intelligence III, L.N. Kanal, T.S. Levitt, R.D. Shachter and J.F. Lemmer, eds. North-Holland, 1989.
- [Binford 87b] T.O.Binford; "Generic Surface Interpretation: Observability Model"; Proc Int Symp on Robotics Research, 1987.
- [Binford 92] T.O.Binford; "Tutorial on invariants and quasi-invariants"; WPAFB, November 18-19, 1992, host Vince Velten; Dr. J. Mundy, GE, "Invariants"; November 18; Dr. T.O.Binford, Dr. T. Levitt; "Quasi-Invariants"; November 19.
- [Binford 93] T.O.Binford and T.S.Levitt; "Quasi-Invariants: Theory and Exploitation"; Image Understanding Workshop, Washington, DC, April 1993.
- [Herskovits and Binford, 1970] A. Herskovits and T.O. Binford. "On Boundary Detection", M.I.T. Artificial Intelligence Lab., Cambridge Mass, AI Memo 183, 1970.
- [Levitt, Binford, Ettinger 88] T.S. Levitt, T.O. Binford, G.J. Ettinger; "Utility Theory in Computer Vision"; proceedings Uncertainty in AI Workshop at AAAI 88; reprinted in Uncertainty in Artificial Intelligence IV, L.N. Kanal, T.S. Levitt, R.D. Shachter and J.F. Lemmer, eds. North-Holland, 1990.
- [Mundy et al 92] J.L.Mundy, P.M.Payton, M.H.Brill, E.A.Barret, R.P.Welty; "3-D Model Alignment without Computing Pose"; Proc DARPA IU Workshop, pp 727-735, 1992.
- [Sato and Binford 92a] H.Sato and T.O. Binford; "On finding the ends of SHGCs in an edge image"; Proc Image Understanding Workshop, 1992.
- [Sato and Binford 92b] H.Sato and T.O. Binford; "BUILDER-I: a system for the extraction of SHGC objects in an edge image"; Proc Image Understanding Workshop, 1992.
- [Ulupinar and Nevatia 90] F. Ulupinar and R. Nevatia; "Recovering shape from contour for SHGCs and CGCs"; Proc Image Understanding Workshop, pp 544-566, 1990.
- [Wang and Binford 93] S.J. Wang and T.O. Binford; "Local Step Edge Estimation: a New Algorithm, Statistical Model and Performance Evaluation"; Image Understanding Workshop, Washington DC. 1993.

Section II

Benchmark

CREATING BENCHMARKING PROBLEMS IN MACHINE VISION: SCIENTIFIC CHALLENGE PROBLEMS

Oscar Firschein*, Martin A. Fischler**, and Takeo Kanade***

*DARPA, **SRI International, ***Carnegie Mellon University

Abstract

We discuss the need for a new series of benchmarks in the vision field, to provide a direct quantitative measure of progress understandable to sponsors of research as well as a guide to practitioners in the field. A first set of benchmarks in two categories is proposed (1) static scenes containing manmade objects, and (2) static natural/outdoor scenes. The tests are "end-to-end" and involve determining how well a system can identify instances (an item or condition is present or absent) in selected regions of an image. The scoring would be set up so that the automatic setting of adjustable parameters is rewarded and manual tuning is penalized. To show how far machine vision has yet to go, a Benchmark 2000 problem is also suggested using children's "what is wrong" puzzles in which defective objects in a line drawing of a scene must be found.

1 Introduction

Speech and natural language researchers at DARPA have made extensive use of a benchmarking approach to obtain a measure of the progress in these fields. This exercise has had two distinct positive effects on the fields. First, since the results of the benchmarks provide a direct quantitative measure understandable by people outside these fields, they have been of great use "politically" within DARPA. Second, through the rigorous comparison among various techniques the benchmarking exercise has spurred advances in the field. This paper discusses the strategy for devising and using a new series of benchmarks in the vision field. We believe that the vision field requires such benchmarking efforts to objectively measure its progress. There have been some previous benchmarking attempts in vision at DARPA and elsewhere, but they dealt mainly with measuring the performance of computer architectures running vision algorithms, rather than with the performance of the vision algorithms and systems.

The goal of creating challenging benchmark problems in machine vision is to pose a reasonably comprehensive

set of vision problems to which proposed advances can be subjected to experimental evaluation. The problems should be formulated in terms of tasks, for a module or for a whole system, independent of any specific techniques - e.g., evaluation of three-dimensional shape recovery rather than evaluation of a shape-from-X method, or evaluation of natural scene understanding rather than evaluation of an expert system-based interpretation system. There would be challenging problems in various categories, e.g., outdoor scenes, manmade objects, time-varying scenes, and so on. After discussing issues and methodologies in creating vision benchmark problems, this paper presents a few example problems in the domain of static natural scenes and in static scenes containing manmade-objects.

2 Previous Benchmarking in Vision

The DARPA benchmark carried out from 1986-89 [1,2] was an attempt to characterize the performance of machine architectures running IU algorithms. As such, it is not directly applicable to the current IU benchmark effort. However, there were several lessons learned, primarily the time-consuming and somewhat expensive nature of the operation.

A more pertinent benchmark approach is the Unmanned Ground Vehicle set of evaluations for all subsystems, including stereo, LADAR, road-following, and path planning. The stereo evaluation, described in these proceedings [3], is of particular interest in this regard. The overall plan was to pursue a three-pronged approach, including analytic models, qualitative "behavioral" models, and statistical performance models. The analytic models are used to estimate the expected depth precision computable with a specific camera configuration. The qualitative models are used to identify key problems for future research. The statistical model is used to produce quantitative estimates of such key factors as the smallest obstacle detectable at a specified distance. Data gathering and preparation required a large

amount of effort. Imagery was collected from five groups; 49 image pairs were selected for analysis and converted to a standard format. An interesting initial result of the evaluation was the identification of the strengths and weaknesses of the various stereo techniques, leading to the possibility of combining them in a system that produces more complete and accurate results than any of the individual techniques.

3 Issues in Vision Benchmark Design

There are a few important issues that arise in vision benchmarking :

- specifying the scope of a problem,
- balancing competitive and collaborative aspects of benchmarking, and
- devising an evolutionary problem selection mechanism for future benchmarks.

3.1 Vision Problem Specification

The critical difference between producing vision benchmarks, and producing those for language and speech, is that the field of machine vision does not yet have widely acceptable specifications for generic problem domains (or representations) on which to base the problem definitions. Further, the number of sample images and supporting data necessary to cover any given problem domain without artificial (unknown) biases seems to be far larger than in language and speech. In language, topics and languages certainly vary a lot; yet a large enough number of news articles, novels, etc., will reasonably cover the problem variations, and text files contain everything the benchmarking algorithms need to employ. In speech, there are variations in frequency, dialects, etc, yet a "numeral digit recognition" problem or a limited vocabulary problem provides some reasonable bound to a problem domain, and a large enough number of speech samples will cover the variations. A high-quality tape of speech (with various types of background noise) is a good universal basic representation for the input data.

In image understanding, the direct analogies do not work as well. Outdoor natural scenes do not seem to have an accessible technical definition, except that people can probably classify a given image as depicting a natural scene or not. A universal representation/media for sensed data describing a scene does not exist. Moreover, the nature of the input devices, the way we acquire images and specify the resolution, the measurable information, etc. can themselves, singly or in combination, constitute major research problems.

3.2 Competitive and Collaborative Aspects

The principal goals of the proposed vision benchmarks are to evaluate scientific progress in specific problem areas, and to make the extent of such progress apparent to the sponsors of the research as well as to the scientists working in this field. Evaluation of a set of alternative solutions to a problem naturally involves comparing the resultant scores and to thus rank the techniques. We can't avoid competition. Making the results of the evaluation difficult to interpret or keeping the identity of the participants secret eliminates the incentive to enter the evaluation and exert the necessary effort to do well.

Nonetheless, it is very important to make sure that we are competing on the right problems, that the competition is fair, and that we don't poison the currently excellent cooperative atmosphere that exists in the DARPA vision research community.

Among its positive benefits, benchmarking will promote collaboration. Many researchers will not be able to afford to develop all the system components themselves in order to enter the evaluation. A module or component that has been proven to have high performance will be transferred from the hands of the developer to other sites whose main research focus is not the module, but rather access to its functionality.

In the specific case of algorithms that are intended to run autonomously, i.e., without manual tuning, it is critical for the purposes of believability that the test data NOT be given to the contestants prior to the benchmark. Further, the problem of automatically finding settings for adjustable parameters (present in almost every vision algorithm) is a key vision problem in which progress should be encouraged - the benchmark could be a positive influence in this regard.

3.3 Evolution of Benchmarks

Since there is enough diversity of opinions about what constitutes the correctness of the output of any component, practical benchmarking tends to be performed on "end-to-end" systems performing a well-understood task. This emphasis on system evaluations can have both positive and negative impacts on the field. Positive effects are: the promotion of research because there exist accepted criteria of progress; the establishment of some priorities on problems to be solved; and an increased awareness of the availability and usefulness of a broader range of techniques for performance improvement. Potential negative effects are: the temptation to use any trick that improves performance on the evaluated task; the premature stifling of new directions in the field, and the reliance on some statistical methods which tends to produce better "average" results. Some of these phenomena, positive and negative, have appeared in the lan-

guage and speech fields since the introduction of benchmarks.

To achieve the positive effects and avoid the negative ones, the vision benchmarks should allow for evolution and expansion as we improve our understanding of the field. This is especially important in vision because vision tasks are not static – they expand. The benchmark problems cannot be a casually controlled ad-hoc collection of problems, or a set of problems carefully tailored for small cliques, each with a special view of how the problem should be solved. If a group of investigators wants to pursue a promising new approach which cannot be evaluated appropriately within the current set of benchmarks, there must be a mechanism to define a new DARPA benchmark if appropriate criteria are satisfied.

4 Deriving Benchmark Problems

4.1 Problem Selection

We must first define the problem domains, such as static outdoor scenes, static scenes of manmade objects, outdoor image scene sequences, and image sequences of manmade objects. A panel will be organized to carefully divide the vision field into categories and subcategories because this categorization is one of the most critical issues in the design of the benchmark. A relatively small number of problems (less than four initially) in each category would be carefully selected by community consensus. These problems should be based on some important vision function, NOT some vision architecture, representation, or technique.

These should generally be retained in the benchmark until they are “solved” or no longer of scientific importance.

The complete benchmark should cover the vision field by defining between five to ten separate problems for the competition; it may be necessary to have two problems in some categories to separately deal with the main dichotomy of strong vs. weak models (e.g., man-made environments vs. natural outdoor scenes). Each problem category may require separate subcategories for different sensing modalities, viewing conditions, and environmental factors; in the case of sensing modalities, the subcategories could be

- intensity images vs range images
- black-and-white vs color (or multispectral)
- significant perspective distortion vs. essentially orthographic projection

The problems listed in Appendix A are a few abstracted versions of possible benchmark entries for the static outdoor scenes and man-made object scenes. They

are offered for discussion in the light of all the above sentiments, but the task of choosing the actual problems still remains. It is hoped that for an initial benchmark a total of no more than four problems will be selected from the set of all submissions. This would allow us to work out the details of the process before an excessive amount of effort is expended.

Finally, it is important to remember that the proposed benchmark will only cover a small subset of the important problems in machine vision. We have not done away with all the traditional methods of reporting and evaluating progress.

4.2 Evaluation Method

Human examiners would select (but not necessarily reveal to the contestants) a few locations in each image that contain obvious instances (item or condition is present or absent) of, for example, the existence of a road or a material like grass or rock. The scoring at each location is binary – correct or incorrect.

Problems, for example, in recognizing natural objects are believed to be difficult enough so that no currently known technique, or brute force approach, can perform well (i.e., within 50% of human performance on the recognition problems and somewhat higher on the geometry problems depending on the availability of calibration data and the nature of the prior models) without additional constraints on the problem (or the provision of auxiliary information, such as manual parameter adjustment to match the given imagery). An obvious advance would be a performance improvement of, say 5 to 10% over that of the previous best known technique. When performance of a computer vision technique reaches (say) 90-95% of human performance, the corresponding problem is considered to have a reasonable scientific solution and further advance is now also considered in terms of engineering criteria (cost, speed, complexity, etc.).

4.3 Competition Procedure

It is intended that there would be a competition once a year to choose the best performing program in some (or all) problem categories. The programs would have to run on specified machine configurations, must take the input images in a specified format, and must produce answers in a specified time interval. To insure an initial reasonable baseline of performance for the most difficult problems, an operator would be allowed to place a specified number of labeled markers in an overlay of the given test image and/or be given (in advance) a small window from the test image to allow system calibration and parameter adjustment. Typical images from each category would be provided in advance, and would not change in nature or difficulty from year to year.

Entry in the competition implies the entrant is willing to make public the theory (and possibly pseudo-source

code) for his algorithms and allow the use of his object code (at least) for scientific purposes.

4.4 Specific Proposal

1. A list of 5-10 problem domains will be selected for the benchmark.
2. A panel of experts would be chosen to define the problems and select the sample and test imagery and the contextual information to be made available. Sample imagery would be available prior to the competition. The actual test imagery would be provided to all interested parties after the competition.
3. The nominal approach would be for the panel to select a few locations in each image that contain obvious instances (item or condition absent or present) of the challenge problem subject matter; this information would not be revealed (for the test imagery) until after the competition; scoring at each location is binary, correct or incorrect.
4. The test could be held yearly (e.g., at the IU meeting or at some selected contractor site) on machines provided or approved by DARPA. Programs must produce answers in a specified time interval. It is intended that the programs will be run without intervention by the contestants, but some provision might be made to allow a contestant to tune his program at a specified penalty to his test score.
5. Theory (and possibly pseudo source code) must be provided in report form, and the compiled code actually used in the competition made available (free, but possibly under license) for scientific use.

5 Conclusion

We have taken the initial steps in developing a new set of machine vision benchmarks in the areas of manmade object scenes and natural scenes. The next steps involve more careful delineation of the experimental protocol, selection of the specific problems, and the gathering of imagery and other test data. We welcome comments on this new DARPA benchmarking effort.

References

1. A. Rosenfeld, "A Report on the DARPA IU Architectures Workshop," Image Understanding Workshop, 1987.
2. C. Weems, E. Riseman, and A.R. Hanson, "A Report on the Results of the DARPA Integrated Image Understanding Benchmark Exercise," Image Understanding Workshop, 1989.

3. R.C. Bolles, H.H. Baker, and M.J. Hannah, "The "JISCT" Stereo Evaluation," in *these proceedings*.

APPENDIX

This appendix offers a set of abstracted versions of possible benchmark entries for discussion in the light of the goals and issues. The task of specifying the actual problems still remains.

A.1 Man-Made Object Scenes

The key issues in setting up the problems for man-made object scenes include:

- amount of clutter in a scene
- amount of occlusion of objects
- class of shapes of objects (e.g., polyhedral vs. curved, planar vs. 3D, fixed vs. articulated or deformable)
- class of surfaces (e.g., textured, specular, diffuse)
- lighting conditions
- kind of imagery (2D vs. 3D, grey-scale vs. color)
- class of transformations applied to object model

Even more important is an evaluation method. We can use the ROC (Receiver/Operator Curve) which plots the false negative rate vs false positive rate as the overall indication of the performance of a system. We should evaluate the accuracy of computed pose as well as the number of free parameters in the system. We should also define a series of increasingly harder problems, such as presented below.

PROBLEM 1: Flat parts (with little or no texture on the parts or background) and known camera orientation. But include significant clutter (e.g. as little as 10% of the features in the image are associated with the object) and significant occlusion (perhaps as little as 25% of the object is visible) as well as noise. The goal is to identify and locate as many instances as possible from a small library of known models. This problem is probably fairly well solved by several existing algorithms. Open issues include how to provide/obtain the models and a range of shapes for each of the models. We can allow the objects to scale, rotate, and translate in the image plane.

Example objects include 2D parts (eg. teletype parts) and tools (wrenches, screw drivers, etc).

PROBLEM 2: Solid rigid objects with no articulation, but with significant clutter and occlusion. Texture is allowed on the objects and background. One version

of the problem would be 3D shape recovery from a single 2D image; a second version could be 3D shape recovery from a 3D image (i.e. from range imagery).

The objects should include a range of shapes and even several shapes that differ only in a few places, so that saliency can be an issue. Examples include models of vehicles or planes, simple office scenes, etc.

PROBLEM 3:

Generic objects. This could include parameterized object classes and articulated objects. The idea is to allow for objects that are nonrigid, while still structured. A tank is an obvious example, given the movable turret. Classes of vehicles in general provide the next level of complexity (e.g. categorizing vehicles as a sedan, a station wagon, a van, etc., as well as localizing it).

PROBLEM 4: Recognition by function. A classic example is a "chair", where the recognition system has to identify objects not only by shape, but also by whether they meet certain functional constraints (such as stable support, a flat surface on which to sit, etc.)

A.2 Static Natural Scenes

1. **Generic (natural) Object Recognition or Classification:** Recognize (point to or delineate) rocks and trees in single images of outdoor scenes. Distinguish between rocks and sand in a desert scene.
2. **Specific Object Recognition:** Recognize (point to, or delineate) the presence of a specific known object in an image (e.g. a particular person). The object (preferably non-rigid) can be partially occluded or seen from any aspect.
3. **Feature Extraction and Delineation:**
 - Extract a road network from an aerial image that can be either vertical or oblique and low or high resolution. For example, the image could even be a view of a partially occluded freeway from a window in a nearby building.
 - Given the skeleton of a tree trunk or tree limb (in a forest scene), accurately delineate the corresponding edges and measure the width of the trunk/limb.
4. **Geometric Recovery From a Single Image of a Natural Scene:**
 - Given a line overlaying an image, determine relative depth (from the camera) along the line.
 - Determine (scene) surface orientation at a given set of points (locations) in an image. The scale of interest will be provided.
5. **Geometric Recovery From Multiple Views Of Some Specified Object:**
 - Model (i.e., recover the 3D geometry) a building in an urban scene from multiple views.

- Render the profile of the skyline seen from a specific ground location, given an overhead stereo pair of a mountain or valley.

- Given a dense sequence of images containing an object of interest (e.g., a tank or a rock) taken from a camera mounted on a moving truck, recover the geometry of the object.

6. **Track a Specific Moving Object:** For example, a specific fish in a fish-bowl containing many fish and other objects that can cause occlusion or temporary disappearance of the fish being tracked. Other possibilities are a person in a crowded store, or a specific tank in a formation moving through a wooded area.

7. **Generic Problems (edge and surface classification) in Image Analysis:**

- Classify specified locations in an image as either edge or not-edge; if edge, then further classify the nature of the edge as either occlusion, orientation, illumination (e.g. shadow), or reflectance edge.

- Classify the material type of selected surface patches in an image as either wood, metal, glass, water, vegetation, sand/rock, brick, soil, sky, cloud, asphalt, or concrete.

- Classify, into say three spectral bands, the color spectrum of selected surface patches in a natural outdoor image.

A.3 Benchmark 2000

Some day we would like vision to be integrated with knowledge and reasoning. A challenge problem to evaluate this capability would require a system to identify objects, use knowledge about these objects and the real world, and apply reasoning to solve a visual problem. We suggest a challenge problem along these lines for the year 2000 using a class of children's picture puzzles in which the child is asked to find "mistakes" in the picture. The problem posed to a vision system would be: Given a line sketch taken from a children's book of "what is wrong" puzzles, see Fig. 1, devise an automatic vision program that can perform as well as a five year old child. The absolute score for any algorithm is the percent of defective objects found in a scene; the relative score can be obtained by comparison with a child's performance.

Section III

RADIUS

RADIUS: Automating Image Analysis Through Model-Supported Exploitation

Shirley J Gee and Arthur M Newman
Artificial Intelligence & Information Systems Department
Hughes Aircraft Company
El Segundo, California 90245-4701

ABSTRACT

Image analysis is a labor-intensive activity that grows increasingly intensive due to the volume of imagery and collateral being collected. Image analysts (IAs) and photo interpreters need to extract accurate yet timely information from the data. Strategically automating portions of the processing will help analysts achieve both objectives, first by eliminating some of the tedium of the activity then by accelerating the process. Model-supported exploitation (MSE) was identified as the technology that will provide this automation. This paper discusses in detail the various MSE design constraints, first as they pertain to the RADIUS problem domain, and then in context of their impact on the design of an MSE workstation.

1. INTRODUCTION

Image processing (IP) and image understanding (IU) have been the subjects of research for many years, both in the academic and the industrial worlds. In 1990, a consortium of MSE experts and researchers surmised that IU technology was sufficiently mature for implementation in an operational capacity. Specifically, IU was thought to be particularly well-suited for application to the problem of image analysis. The IA community was targeted as the end user of this technology, and the MSE concept was designed to address its needs.

The goal of MSE is to support IAs in their work by using three-dimensional site models to produce visual aids and provide geographically referenced collateral information. The design of an MSE workstation must take into account the following three constraints:

- [1] Operational needs
- [2] State of technology
- [3] Cost and development schedule

Fully-automated processing has often been

touted as the theoretical goal, but the desirability of this is questionable since the ultimate use of the end product is of such import that skilled IAs will always be needed to verify the processing results. Furthermore, technology that is both robust and reliable enough to achieve this objective is currently not realistic. Practical development of an MSE workstation also requires adherence to budgetary and scheduling constraints.

1.1. Image Analysis

Four basic activities were identified as tasks for which IAs are responsible:

- [1] **Change detection** - the location and identification of significant changes in an image, both physical (e.g., construction) and logistical (e.g., an increased number of vehicles in a delineated area)
- [2] **Negation** - the determination of when a change first appeared at a site
- [3] **Detection and counting** - the counting of all instances of specified objects regardless of quantity, location, or orientation in an image
- [4] **Trends and history** - or trend analysis, the chronological documentation of events at a site

Based on an analysis of current IA activities, it was hypothesized that all image analysis activities consisted of some combination of these four tasks. In addition, the consortium identified four tools for supporting IAs in their work:

- [1] **Registration** - the establishment of a mathematical point-to-point mapping between two data types (e.g., images, site models, maps) to correlate key objects, regardless of collection source
- [2] **Perspective, geometric modeling, and orientation (PG&O)** - the two-dimensional or three-dimensional rendering of data types (e.g., images,

- site models, maps) to produce visual aids
- [3] **Recognition guides** - digitized image chips, drawings, and text that assist IAs in identifying objects
- [4] **Interpretation aids** - site models and recognition guides for non-literal image analysis

Due to the present manual nature of image analysis, these tools do not currently exist in the operational environment. Consequently, their impact on IA productivity remains to be validated. Finally, two databases were identified as being part and parcel of the MSE concept:

- [1] **Site baselines** - the primary source of site specific information (e.g., equipment normalcy ranges, functional area layouts, historical information)
- [2] **Site folders** - contain other collateral data specific to a site (e.g., reference imagery, maps, textual reports)

Site baselines and folders currently exist in hardcopy form. These ten items are collectively known as the RADIUS application concepts. The design for an MSE workstation must provide interactive tools for addressing each of these application concepts, either in part or in whole.

1.2. Definition of MSE

The MSE concept is grounded in the existence and exploitation of site models for image analysis and consists of two components:

- [1] The development and maintenance of site models using IU techniques
- [2] The use of site models for image exploitation tasks, either directly by IAs or via a set of model-based IU exploitation tools

The IA community has expressed great enthusiasm for the exploitation component in general and for the exploitation tools in particular. IU researchers recognize, however, that an efficient site modeling capability must exist for automation of exploitation to be possible.

1.3. User Concerns

Productivity, accuracy, and timeliness in

image analysis are issues of fundamental importance to the IA community. Two government-supported projects are specifically geared toward addressing these issues. They are the Research and Development for Image Understanding Systems (RADIUS) project and Workstation 2000.

1.3.1. RADIUS

RADIUS is planned to progress in two phases. The goal of Phase 1 was to characterize the state of technology and to define the MSE operations concept, resulting in the design of a testbed system. Phase 2 is geared towards testbed development and evaluation in an operational environment. In total, RADIUS is scheduled to run five years, with Phase 1 being a two-year exercise and Phase 2 being a three-year endeavor.

The RADIUS Phase 1 contract was awarded to Hughes Aircraft Company in mid-1991, with BDM Federal, Computing Devices International (CDI), the Hughes Research Laboratories (HRL), and the University of Southern California (USC) forming its subcontracting team. Hughes, HRL and USC formed the IU technology assessment subteam, while BDM and CDI worked on the MSE concept definition. USC offered consultation and insight into continued MSE research. The efforts of the Hughes team resulted in the conclusions summarized in this paper. The contract is scheduled to end in June 1992, with the Request for Proposal for Phase 2 being issued in late July.

1.3.2. Workstation 2000

Workstation 2000 is an umbrella concept that covers the gamut of IA concerns regarding implementation of MSE in an operational environment. One concern is that any MSE workstation designed to function in an operational capacity must exploit existing collateral, that is, the workstation must be integrated with the databases currently available to the IA community. Another concern is that the workstation tools must facilitate image analysis as defined by IAs, that the tools be perceived as unobtrusive and non-invasive. A third concern is the need for speed, the desire for timeliness as well as accuracy. These concerns were primary factors in the MSE workstation design.

2. OPERATIONAL NEEDS

The two operational components of an MSE system are the modeling subsystem and the exploitation subsystem. The modeling subsystem provides a means for generating the three-dimensional site models and for linking collateral data to these models. The exploitation subsystem provides the visual aids that support IAs in their activities. A series of experiments were conducted to determine what IAs wanted from site modeling and exploitation tools. The requirements the analysts placed on the subsystems were used to derive the MSE operational needs.

2.1. The Modeling Subsystem

Informal site models are currently being used by IAs. These include physical entities (e.g., sketches, NEL-produced diagrams) as well as the knowledge an analyst internalizes as he familiarizes himself with a site. For a site model to be useful to an MSE workstation, however, all pertinent data must be stored explicitly. Furthermore, related information (i.e., collateral data) must be easily accessible.

A site model is a three-dimensional, georeferenced wireframe representation of the objects at a site. Included are both physical (e.g., buildings) and non-physical (e.g., functional areas) objects, their attributes (e.g., labels), and links to the collateral data that is associated with the particular site. While it is obvious that site models must include objects which are of analyst interest, it is equally important that they include objects which are of processing utility. By definition, then, site models contain more components than an analyst would want to see or an algorithm could be designed to process.

The appropriate level of model detail is an issue over which IAs differ and for which the capabilities of IU vary. For example, an analyst who is primarily interested in detecting gross changes (e.g., demolition of a building) may be satisfied with a simple block diagram, while one who is interested in the function of a building may require that substructures (e.g., doors, windows) be included. The modeling subsystem must therefore be capable of generating models at a variety of detail levels to support the variety of IA needs.

Another issue of concern is the question of which objects are necessary and sufficient for a site model. Just as the appropriate level of model detail was a function of the exploitation goals of the IA, the nature of the objects in which the IA has a particular interest are likewise defined. To support the variety of disparate interests without inundating any particular IA with superfluous information, the concept of layers was introduced. Site model layers are subsets of the site model and reflect either community-wide standards (e.g., the baseline layer) or individual interests (i.e., user layers). Access to the various layers may be global or privileged.

From the user perspective, one of the most important conditions for site models to be considered useful is that they are kept reasonably up-to-date. Thus, in addition to the tools for creating the initial model, the modeling subsystem must also provide a mechanism for site model update: for creating new objects, modifying existing objects and their collateral links, or deleting destroyed objects. The MSE system must be capable of maintaining the change history of a site to enable IAs to trace the evolution of the site.

2.2. Exploitation

Those IAs interviewed during the RADIUS Phase 1 experiments summarized their exploitation responsibilities in the following sequence of activities:

- [1] Prioritize the imagery to be exploited
- [2] Locate the site on the image
- [3] Take a quick look around the site
- [4] Detect and count things of interest
- [5] Look for site activity
- [6] Look for site changes
- [7] Discuss findings with other IAs
- [8] Take a final look around the site
- [9] Report findings and conclusions
- [10] Retask the site if necessary

While these activities seem ill-correlated with the four basic IA tasks described in Section 1.1, upon further consideration, it will be seen that the fundamental analysis process, that of assessing the existence, magnitude, type, and timing of change, is reflected in both sets of task descriptions. What is important to note is the IA's need to validate and verify change personally. In

essence, then, the exploitation subsystem must provide the IA with interactive tools for making these assessments. The subsystem must also offer mechanisms for generating reports of IA findings.

3. TECHNOLOGY ASSESSMENT

The underlying assumption is that IU technologies will be available in the RADIUS Phase 2 timeframe to fulfill the MSE operational needs as outlined in the previous section. IU systems were assessed during Phase 1 to project realistic automation levels for the various MSE tools in the Phase 2 testbed system. The assessment focused on technologies that addressed the following application concepts: site model construction and update, registration, PG&O, and site baselines and folders.

3.1. Technology Goals

Any MSE technology targeted for inclusion in the Phase 2 testbed system must have the basic objective of lightening the IA's workload. Because IAs have exploited imagery without IU tools for years, successful introduction of MSE to the IA community depends largely on the cultivated perception of MSE as an aid rather than a hindrance. With this caveat, automation levels (i.e., manual, semi-automated, or automated) and the relevant user interface issues were found to be useful performance measures.

The primary issue with manual tools is the user interface. The tools must provide sufficient visual feedback to the IA so as to achieve the required accuracy. Such tools must permit rapid, intuitive operation without overwhelming the user with options. For semi-automated tools, the issue expands to include real-time response. The objective in this case is to provide accurate results in less time than would be needed by an IA using manual tools. For automated algorithms, run-time, accuracy and false alarms become key concerns.

Automated exploitation algorithms face far more stringent requirements than those for site model construction, since exploitation is often a time-critical activity that demands quick turnaround. Unfortunately, poor accuracy and high false alarm rates are often obtained at the price of speed. These errors undermine the value of automation by forcing the IA to compensate for the

inaccuracies. Since site model construction is less likely to be bound by such tight timing constraints, modeling algorithms can be designed to emphasize accuracy over speed.

3.2. Characterization Methodology

An IU system qualified for Phase 1 characterization if it satisfied one of two conditions:

- [1] It was a commercial product
- [2] It was the subject of active research

The literature is replete with descriptions of systems that address elements of the RADIUS problem domain. Rarely, however, have the algorithms been subjected to extensive testing. Success of the Phase 2 testbed system therefore depends on the availability of the original developers to support integration, bug fixes, and system enhancements. While other highly relevant systems have been presented in the literature, it is doubtful whether they would be ready for presentation to an IA within the RADIUS Phase 2 timeframe.

The last RADIUS report¹ outlined plans to provide test sets of imagery to system developers, along with ground truth and code for initial on-site characterization. Generating representative sets of unclassified image data and having developers characterize their work without compensation have been inordinately time-consuming. Consequently, this paper reflects a mixture of formal characterization results mixed with an internal assessment of these systems and the techniques they employ.

A decomposition of the MSE applications was presented in the RADIUS Technology Development Plan². That decomposition was used as the framework for characterizing the functionality and the availability of the relevant IU systems. Near-, mid- and long-term availability correspond to the three years of RADIUS Phase 2: 1994, 1995, and 1996. The following subsections offer a conservative estimate of the technology available to support MSE. It is not a recommended architecture for Phase 2, but rather a response to the assumptions regarding IU maturity and the degree to which IU can satisfy MSE operational needs.

3.3. Site Model Construction

Figure 1 shows the decomposition of site model construction and the technologies as they are projected to be available during Phase 2. For example, during the first year, *Imagery Selection* is supported by cloud detection, which allows the system to select only those images offering a clear view of the site. *Image-to-Image Registration* can be implemented via SMED (Hughes) or MAIR (Harris Corporation). SMED takes advantage of all available camera parameters and other system specific data, relieving the IA from having to select more than two pairs of tie points. As tested against the RADIUS characterization test set, MAIR automatically registered image pairs with a mean Y-disparity of less than 2.5 pixels. *Terrain Extraction* is automatically performed by GLSM (GDE Systems Incorporated - GDE), which is being incorporated by the Defense Mapping Agency (DMA) into their mapping workstations this year. All these systems were designed to work on classified imagery and their associated collateral.

Several options exist for *Object Modeling* in the near-term. For manual modeling, three systems offer broad support for the various MSE applications: the RADIUS Common Development Environment³ (SRI/GE), GLMX (CDI), and SOCET SET (GDE). RCDE is based on SRI's Cartographic Modeling Environment, a manual site modeling and IP package. GLMX was the basis of the RADIUS concept validation experiments and has been used in a classified operational environment to rapidly generate site models for bomb damage assessment. SOCET SET is a commercially available manual site modeling package that draws on GDE's experience with DMA mapping workstations.

These same three sources are addressing near-term semi-automated *Object Modeling*. SRI, as a RADIUS-related Broad Area Announcement (BAA) awardee, is extending its model-based optimization technology⁴ to extract roads, railroads and buildings. CDI has developed a means of using stereo imagery to accurately refine coarsely-specified three-dimensional wireframe

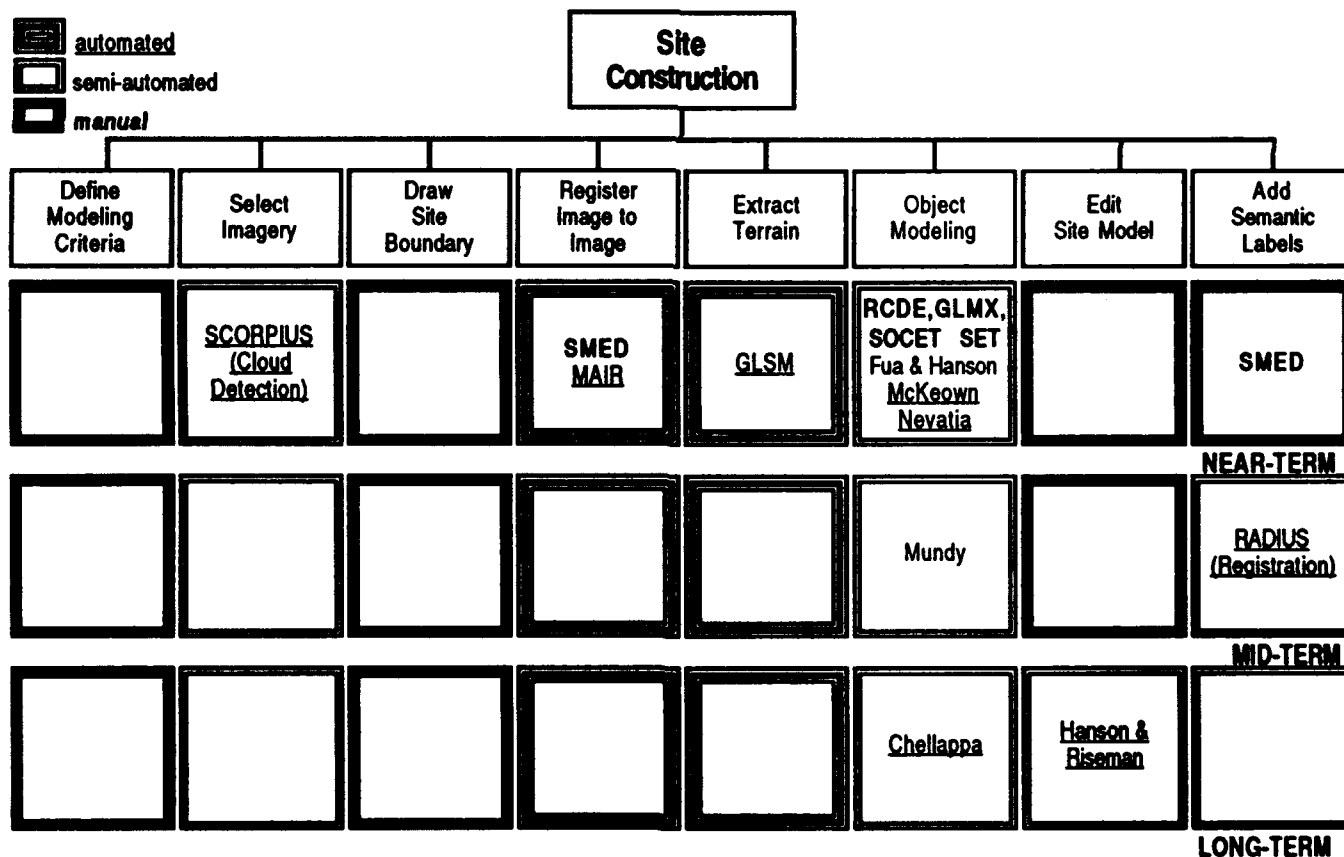


Figure 1. Site Model Construction. Each algorithm is coded according to its automation level. Similarly coded are the boxes extending below each task, reflecting the automation level of that task for each year of Phase 2. Tasks for which no technology is assigned are handled manually via database and other non-IU technologies.

primitives. GDE has enhanced SOCET SET with algorithms that track roads and railroads and locate rooftops in stereo imagery given an initial user estimate. The constraint-based semi-automated algorithm developed at GE/CRD⁵ is estimated to be a mid-term capability. The specification of model primitives for use by the GE/CRD system may require more mathematical expertise than possessed by a typical IA, thus requiring a library of ready-made primitives or a simplified user interface.

Near-term automated *Object Modeling* is likely to be available as systems from USC and Carnegie-Mellon University (CMU). USC has two approaches to automated structure extraction, one based on stereo⁶ and the other on shadow evidence⁷. The CMU offering is based on the BABE⁸ system. Each of the three systems is being enhanced to handle oblique imagery. These fully-automated systems have been rated as near-term technologies with the understanding that they operate with an acceptably low false alarm rate. The long-term automated building extraction algorithm being developed by the University of Maryland (UMd) under a BAA extracts evidence from stereo imagery and uses a truth maintenance system to search among the building hypotheses generated⁹.

The manual site modeling systems mentioned above are applicable to the *Edit Site Model* task, in which the user may refine, add or delete elements of the model. A model extension capability is being developed at the University of Massachusetts, Amherst (UMass) under a BAA. Positional accuracy of site model elements will be refined automatically via induced stereo generated over multiple overlapping views of the site. This algorithm is considered a long-term capability.

A body of two-dimensional campus-style maps, to which collateral information is attached, is currently available to IAs. Buildings, parking areas, points of entry and other intelligence-related site features are typical components of these maps. More importantly, the maps include labels and identifiers keyed to the various map elements. Registration of this map to imagery of the site facilitates automated entry of these labels and identifiers into the generated site model. After projecting the campus-style map into the geometry of an image of the site, SMED offers a near-term manual capability for aligning the map with the

imaged site. An automated map-to-image registration is considered a mid-term capability.

3.4. Site Model Update

Many of the tasks and technologies for *Site Model Update* are shared with *Site Model Construction*. Consequently, Figure 1 also applies to site model update. The primary difference is the availability of the site model as a starting point in *Site Model Update*. New images are registered initially to the projected site model, instead of to other images. SMED provides a "drag-and-drop" near-term manual registration capability. As is the case with map-to-image registration, an automated model-to-image registration capability can be developed for mid-term testbed inclusion.

3.5. Site Baselines and Site Folders

Site baselines are textual descriptions that are updated annually or as frequently as needed. They document normal activity and highlight specific areas of interest. Site folders contain reference data (e.g., maps, charts, select images) that aid IAs in exploiting imagery of the site. IU support for baselines and folders comes in the form of model-to-reference data registration (Figure 2).

3.6. Exploitation Tasks

RADIUS Phase 1 did not include the four basic image analysis tasks (Section 1.1). While Phase 2 will explore these application concepts more extensively, change detection has been identified as a BAA research area. Figure 3 shows the preliminary requirements common to all exploitation: *Determine Image Utility* and *Register Image to Model*. The combination of cloud detection and model-to-image registration helps automate the image prioritization process, thereby streamlining the exploitation process. UMd is developing a semi-automated change detection capability, scheduled for completion by the third year of Phase 2.

4. TECHNOLOGY STUDIES

As shown in Section 3, registration is fundamental to the success of RADIUS. Several data types must be registered to one another (i.e., imagery, maps, models). While pieces of these registration tasks have been addressed, most

notably image-to-image, there are aspects of the problem specific to the proposed operational environment and data. To validate the concepts that would allow timely and accurate automated registration, the Hughes team undertook two studies. The first relates to image-to-model registration, specifically the registration of large site models to large images. The second study investigated the utility of the campus-style collateral maps mentioned in Section 3.3.

4.1. Registration of Large Models to Large Images

The problem of matching three-dimensional models to images is similar to object recognition where an object in an unknown position and orientation is recognized. There are, however, significant differences. The site model can consist of several hundred objects, some of which may lie off the image. The model is constrained to lie on a known ground plane, leaving only one unknown orientation parameter. Since scale is given, there are only two degrees of freedom for position. Matching is further restricted by *a priori*

knowledge of the camera parameters.

Since the site model is very large, the first step in registration is to decompose the site model into distinctive objects. By using only the most distinctive objects in the model-to-image matching process, computational expense is reduced significantly. After the objects are matched to features extracted from the image, the camera parameters are then updated.

The registration system was developed using unclassified data of Fort Hood. CDI provided camera models and a site model containing 148 buildings. Two images, one nadir (FH1) and one oblique (FH2), were used in the experiments. FH1 was used to build the site model; FH2 was not. The image data was scaled from 16 to 8 bits per pixel, and the images were subsampled by a factor of four. Since the camera models were fairly accurate in overlaying the site model onto the image, two types of error were introduced to test the system. Translation error was introduced by translating the projected model in the image plane. Model error

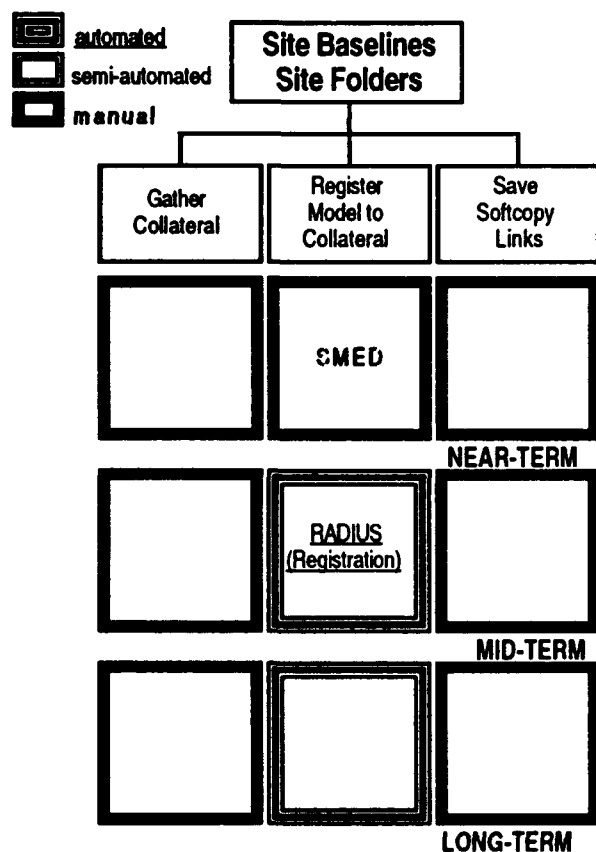


Figure 2. Site Folders and Baselines. Multi-source registration is vital to site baselines and folders.

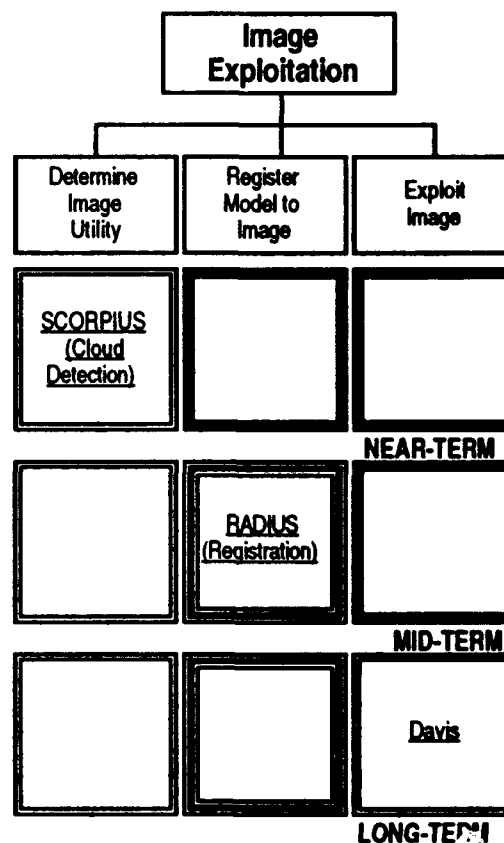


Figure 3. Exploitation. Exploitation depends on clear imagery and accurate model-to-image registration.



Figure 4. Registration of Fort Hood Site Model to FH1. The registration technique yielded a robust solution.

was introduced by randomly stretching each side of a building in one dimension. The latter error did not reflect true modeling error, but it did provide a means of testing the effect of noise in the model. Several tests were run using different error values. Out of the 148 buildings in the site model, 20 distinctive buildings were chosen for matching. Volume, area, and shape as defined by the total number of roof vertices were used to compute distinctiveness. Building locations were not

considered, although they turned out to be well distributed spatially. USC used extracted and extended lines to compute junction features. Figures 4 and 5 show the full registrations of FH1 and FH2. Figure 6 shows an enlarged portion of Figure 4.

A Hough transform was used to compare the locations of predicted and extracted junctions over the search window. Knowledge of the translation error was used to restrict the parts of

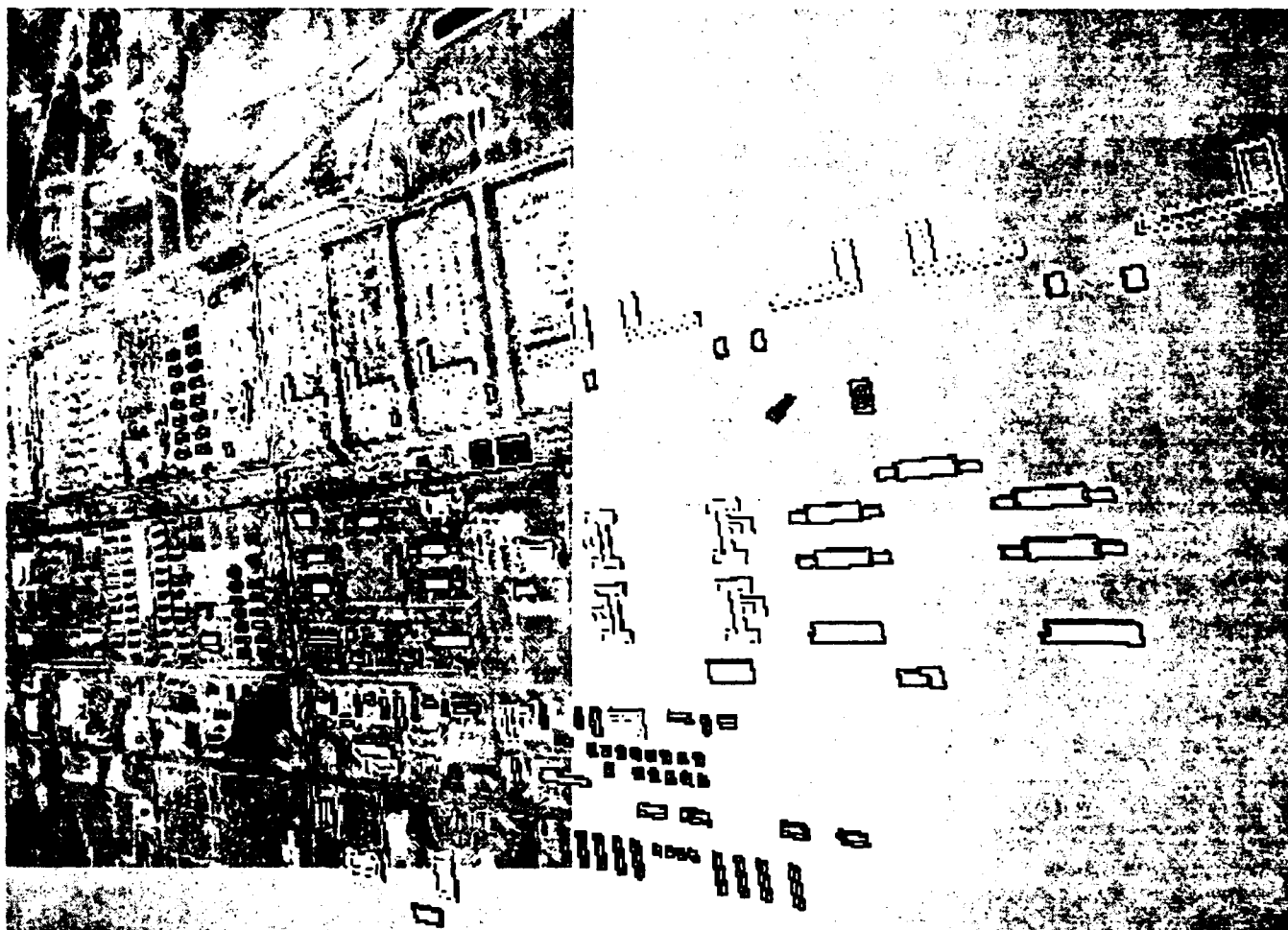


Figure 5. Registration of the Fort Hood Site Model to FH2. The twenty most distinct buildings are highlighted in white. Half of them extend beyond the image boundary.

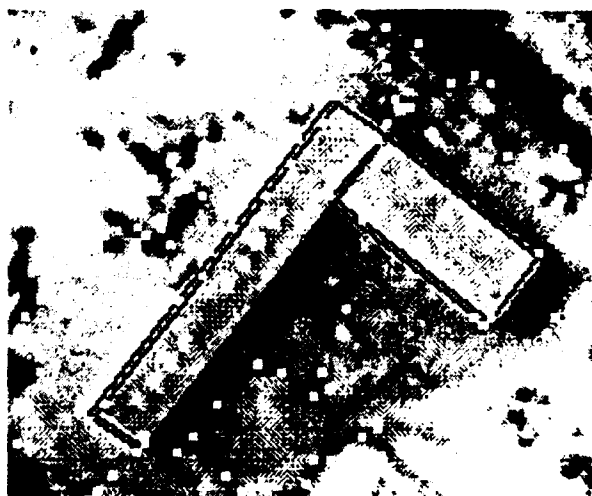


Figure 6. A Building in FH1. The site model overlay is black, the extracted corner junctions white.

the image that were processed and to restrict the neighborhood around the projected model corner in

which image junctions were matched. Results were described by the estimated translation error and by the number of junctions found at that translation.

Count	Count/Peak	T
165	1.00	(301,250)
85	0.52	(68,349)
84	0.51	(371,336)
75	0.45	(313,275)
73	0.43	(73,345)

Figure 7. Accumulator Array for First Test Using FH1. The maximum for the correct peak (165) is nearly double that of the next closest estimate (85).

In the first test using FH1, no error was added to the site model, but a translation error of $T=(300 \text{ rows}, 250 \text{ columns})$ was introduced. The search window in which the model was expected to lie was 400×400 pixels, effectively 1600×1600 pixels in the original image. Figure 7 shows the top

five entries in the Hough accumulator array indicating the matched junction count, the count-to-peak ratio, and the estimated translation for the first test.

In the second test, each side of each building in the model was stretched by 10 pixels, while T was held at (0, 0). As before, the search window was set at 400 x 400 pixels. Once again, the peak was very strong (Figure 8).

Count	Count/Peak	T
180	1.00	(1, 1)
137	0.76	(-5, 5)
104	0.58	(331, 155)
94	0.49	(-137, 108)
89	0.49	(-184, -28)

Figure 8. Accumulator Array for Second Test Using FH1. A robust solution is generated even in the presence of model inaccuracies.

The same two tests were performed on FH2. Inherent discrepancies were known to exist between the site model and the image since FH2 had not been used to generate the model. Consequently, smaller errors were added. In the first test of FH2, no error was added to the site model and T = (-100, -150). The search window was 200 x 200 pixels, effectively 800 x 800 pixels in the original, unreduced image.

Count	Count/Peak	T
56	1.00	(-29, -26)
50	0.89	(-100, -149)
47	0.84	(-154, -194)
46	0.82	(8, -102)
45	0.80	(22, -112)

Figure 9. Accumulator Array for First Test Using FH2. A sharp peak is not produced, since half of the distinct elements extend past the image.

In this example, the estimated translation error closest to the actual translation, (-100, -149), was not indicated by a strong peak, unlike the examples of FH1 (Figure 9). Nevertheless, it was within the top five maxima. As shown in Figure 5, a large number of distinct buildings used for matching lie off the image, increasing the likelihood of false matches.

In the second test of FH2, each side of each building in the model was stretched by 10 pixels in a single direction, while T was held at (0, 0). The

search window was again set at 200 x 200 pixels. As in the first test, the estimated translation error closest to the actual translation, (1, 0), was not indicated by a strong peak (Figure 10).

Count	Count/Peak	T
84	1.00	(-99, -196)
82	0.98	(-38, -63)
80	0.93	(1, 0)
78	0.89	(-126, -196)
75	0.83	(-58, -109)

Figure 10. Accumulator Array for Second Test Using FH2. Although not indicated as the best solution, the correct translation was within the top five.

These experiments demonstrate that model decomposition followed by a simple matching procedure is a promising technique for registering large site models to large images. Decomposition reduces the search space, since only distinctive buildings need to be matched. Decomposition can be improved by including spatial information. One approach is to treat closely-situated, similar structures as non-distinctive. A second tact is to identify distinctive patterns of structures and to later match them as a unit. Finally, additional processing is required in cases where only a portion of the site is imaged (e.g., as in Figure 5).

4.2. Registration of Campus-Style Maps to Imagery

Registration of campus-style maps to imagery is the problem of registering two-dimensional site models to imagery. This problem was solved for two scenarios on the SCORPIUS program. The goal of SCORPIUS (the Strategic Computing Object-directed Reconnaissance Parallel-processing Image Understanding System) was to demonstrate automated exploitation of aerial imagery. Registration on SCORPIUS used image acquisition parameters to project the site model into the image plane then matched pre-computed tie points to features extracted from the image. The problem was one of determining the rotation and translation between the projected model and the underlying data.

The registration system was developed using a campus-style map and imagery. Two oblique images of a site were selected, one reflecting summer conditions (e.g., clear weather) and the other showing winter conditions (e.g., haze and snow). The map was scanned into softcopy and

stored as a semantically-labelled two-dimensional site model. For study purposes, tie points were determined and labelled manually. The site model was projected by the government-furnished camera simulator into the geometry of the selected images. Using SCORPIUS software, the map and its related collateral were registered successfully to the two images.

The hypothesis is that information from the registered map will provide valuable *a priori* knowledge to subsequent IU algorithms. More specifically, object modeling systems will have access to cues regarding the number, location, orientation, size, and footprints of important objects in an image. As a result, automation of the site modeling process is simplified, being modified from a general to a directed search problem. This study demonstrated the viability of using pre-existing campus-style maps to obtain these cues.

5. CONCEPTUAL DESIGN

The conceptual design for an MSE workstation must satisfy the operational needs enumerated in Section 2 while taking into account the technological constraints outlined in Section 3. The critical concerns for IAs are those of confidence in an accurate site model and in the results of automated processing. The key caveats issued by the IU technologists focus on the boundary conditions and accuracy rates of their algorithms.

5.1. Operational Concept

The success of an MSE workstation design is grounded in the integrity and usability of the underlying site models, as required by both the IAs and the IU algorithms. The initial emphasis had been on automation as the end goal in workstation technology. In crafting a realistic operations concept, however, the Hughes team found that IAs questioned the desirability of full automation. Analysts believe that ultimate responsibility for the processing results was still theirs, thereby making a more interactive, checks-and-balances system more appealing.

Technology characterization (Section 3) validated the IAs' concerns, showing that systems which were said to be fully automated still fell short of meeting operational requirements. To accommodate the wishes of the IA community and

the limitations of existing technology, the MSE operations concept includes three levels of automation: manual, semi-automated, and automated, with manual capabilities existing as a fallback position to ensure system functionality under all operating conditions. The RADIUS technology studies (Section 4) demonstrated that through judicious combination of existing data and technology, such an operations concept is a reasonable and realistic objective.

5.2. Technical Requirements

The technical requirements for the MSE workstation stem from the functional flows shown in Figures 1, 2 and 3. Although the flows for site modeling/update and exploitation were conceptualized prior to IA consultation, the RADIUS Phase 1 experiments proved the viability of these designs. The technical requirements that support these functional flows in three levels of automation focus on user interface issues and data exchange standards.

A fundamental characteristic of a workable workstation design is a minimization of the difference between the output of automated processing and any manual effort. The end product, whether it be the site model itself or an exploitation report, must have a standard look-and-feel no matter which portion or how much of the product was generated by IU or a human analyst. A significant implication of this requirement is that the various processing algorithms must work on components of the modeling or exploitation problem that are intuitively self-contained to facilitate human interaction. Having multiple options for each processing stage also mandates the formal definition and strict enforcement of data formats to ensure easy system integration and upgrade.

6. CONCLUSION

As with any system supported by research, there is a tension between the needs of the user community and the capabilities of the underlying technology. By including multiple automation options in conceptual design for an MSE workstation, an attempt has been made to provide the user with ultimate control over the system while giving automation every opportunity to demonstrate its effectiveness. The insistence upon

user and data interface standards serves to simplify system development and integration.

7. ACKNOWLEDGEMENTS

Related research activities have been supported by the Data Systems Directorate of Hughes Aircraft, El Segundo, California, and Hughes Research Laboratories, Malibu, California.

The authors would like to thank the following individuals for helping to develop the MSE conceptual design: Judy Bailey, Jeff Edwards, Mike Kelly, Mike Murphy, Ram Nevatia, Randy Onishi, Donald Query, Jan Sargent, Jim Semrad, Teresa Silberberg, Mark Sleeth, Ron Tajii, Felicia Vilmrotter, and Chris Winicki.

Special thanks to Teresa Silberberg for documenting and contributing the results of the Fort Hood registration study and to Sharon Linderman for editing this paper and ensuring its readability.

8.0. BIBLIOGRAPHY

- [1] J. Edwards, S. Gee, A. Newman, R. Onishi, A. Parks, M. Sleeth and F. Vilmrotter, "RADIUS Research and Development for Image Understanding Systems Phase 1," Proceedings of the DARPA Image Understanding Workshop, San Diego, January 1992.
- [2] Hughes Aircraft Company, "RADIUS Model-Supported Exploitation Technology Development Plan," El Segundo, March 1992.
- [3] J. L. Mundy, R. Welty, L. Quam, T. Strat, W. Bremner, M. Horwedel, D. Hackett and A. Hoogs, "The RADIUS Common Development Environment," Proceedings of the DARPA Image Understanding Workshop, San Diego, January 1992.
- [4] P. V. Fua and A. J. Hanson, "Extracting Generic Shapes Using Model-Driven Optimization," Proceedings of the DARPA Image Understanding Workshop, Washington, April 1988.
- [5] V. Nguyen, J. L. Mundy, and D. Kapur, "Modeling Polyhedra by Constraints," Proceedings of the DARPA Image Understanding Workshop, San Diego, January 1992.
- [6] C-K. R. Chung, *Deriving 3D Shape Descriptions from Stereo Using Hierarchical Features*. PhD thesis, University of Southern

California, November 1992. IRIS Technical Report 302

- [7] A. Huertas and R. Nevatia, "Detecting Buildings in Aerial Images," Computer Vision, Graphics and Image Processing, Vol. 41, No. 6, pp. 131-152, 1988.
- [8] R. B. Irvin, and D. M. McKeown Jr., "Methods for Exploiting the Relationship Between Buildings and Their Shadows in Aerial Imagery," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 19, No. 6, pp. 1564-1575, September 1990.
- [9] V. Venkateswar and R. Chellappa, "A Framework for Interpretation of Aerial Images," Proceedings of the International Conference on Pattern Recognition, Vol. 1, pp. 204-206, Atlantic City, June 1990.

Model Matching and Extension for Automated 3D Site Modeling

Robert T. Collins, Allen R. Hanson, Edward M. Riseman, Yong-Qing Cheng

Department of Computer Science
University of Massachusetts
Amherst, MA. 01003

Abstract

This paper presents an overview of a system for site model extension being developed for the Radius (Research and Development for Image Understanding Systems) project by the University of Massachusetts. Automatically registering imagery to a geometric site model, and extending that model to include new features of interest seen from multiple views, represents an important application of image understanding technology to the task of model-supported image exploitation. The completed system will contain modules for performing feature extraction, model matching, pose determination, and triangulation.

1 Introduction

Acquiring accurate 3D site models from a set of images is a difficult task. Due to the ambiguity inherent in the projection of three-dimensions down to two, general structure recovery is not possible without additional constraining knowledge. We begin by assuming that a partial scene model is available and that for each image the internal (lens) and external (pose) parameters of the camera are approximately known. Our models consist of sets of points, lines, and planes, represented in a three-dimensional world coordinate system. The model is partial in that we do not assume all the important features of the scene are modelled. It is not necessary that the features be connected, confined to a single object, nor is it even necessary that they be known precisely. In this sense we use the term 'model' in a much broader way than is commonly used in the graphical modelling community, although our definition covers the traditional usage as well and thus includes wire-frame and surface-based models.

Given an initial partial model, and a set of images of the site, our goal is to extend the model to include previously unmodeled site features (model extension), and to reduce the inaccuracies in the existing model

(model refinement). This process can be repeated as new images become available, each updated model becoming the initial partial model for the next iteration. Thus, over time, the site model will be steadily improved to become more complete and more accurate.

The process of model extension and refinement can be broken into four important subtasks. First, feature extraction routines are run on new images to reduce the vast amount of incoming data into a manageable set of symbolic geometric descriptions. Second, a model matching procedure uses an initial guess of the camera lens and pose parameters for each image to guide its discovery of the correspondence between model features and extracted image data features. These correspondences are used to perform a resection of the pose parameters for each camera, to find a more accurate description of the transformation between the three-dimensional site model and each two-dimensional image. The updated transformation parameters enable a final process of multi-image triangulation to determine the position of new features in the model coordinate system, and to update the positions of old features based on the new image data. Images of significantly disparate viewpoint can yield a quite large baseline, resulting in very accurate reconstructions.

1.1 The role of the imagery analyst

The imagery analyst (IA) plays a key role in the system being developed. Early versions will require IA guidance at several stages. Most obvious, the initial partial site model we require has to come from somewhere, and the most likely provider in the short term is the IA. Since our models are required to be merely a collection of geometric primitives known in three-dimensions, considerable flexibility is available in their specification. The model may consist of nothing more than the known 3D locations of a set of visually distinctive scene points. Alternatively, the IA could fabricate wire-frame 'boxes' of the appropriate dimensions and locations to fit several significant buildings at the site. Since model extension locates new scene features with

*This work was funded by the RADIUS project under DARPA/Army contract number TEC DACA76-92-R-0028 and by DARPA/TACOM contract DAAE07-91-C-R035.

respect to preexisting ones, the best results are obtained when initial model features are spread out over the whole area of interest, both in terms of ground positions and in height. If enough initial features are given, none of them needs to be specified very precisely, since they will be automatically refined later in the model extension process.

Another important piece of information that will need to be provided to the system is an estimate of the camera lens and pose parameters for each image, which provides the system with an initial estimate of the transformation mapping three-dimensional site features into observed two-dimensional image features. This estimate is used to build the appropriate geometric template for matching the initial site model to extracted image features, and to restrict the search for feature correspondences to a manageable set of likely candidates. Some subsets of parameters may be known beforehand, or recorded when the picture is taken. For example, it is customary to assume the internal camera lens parameters are known and remain constant for multiple images taken by a single camera. Other parameters related to camera pose such as altitude or look angle may be measured at the time the photo is taken, and are provided with some estimate of their uncertainty. If at the time of interpretation some parameters still have no recorded values, an estimate will need to be provided by the analyst. It should be stressed that the initial camera parameters supplied to the system do not have to be accurate enough to provide precise 3D triangulation, but only good enough to allow model matching to find the correct correspondences between model and image features with reasonable computational cost. These estimates will be refined via camera resection following the determination of model to image feature correspondences.

One final area where analyst interaction can greatly benefit the model extension process is in selecting which new image features are interesting enough to be worth adding to the model. The automated system could conceivably be turned loose to add every feature for which a correspondence is found across at least two images – the result would be analogous to a digital terrain model, being refined to finer and finer resolutions as new images are processed. While this may be exactly what is wanted for some tasks, it would not be particularly meaningful for interpreting urban and industrial sites, where one might like to structure the developing model in terms of high-level concepts such as 'road' and 'building', and thus derive additional detail only for particular areas of the model.

1.2 Paper overview

The remainder of this paper describes, section by section, each of the four stages of model extension: feature extraction, model matching, camera resection and triangulation. The aim is to present a brief overview

of the entire process, rather than an in-depth analysis of any one piece. Two of the components, feature extraction and model matching, are currently under evaluation on the model board images, and illustrative examples are included in those sections.

Future research will focus on ways to further automate the model extension process. For example, we hope to acquire initial partial site models from scratch. This process requires determination of correct feature correspondences across multiple images. When accurate initial estimates of the camera parameters for each image are available this is indeed feasible. Otherwise, the very difficult problem of finding correspondences between two images taken by different cameras, possibly from significantly different viewpoints, must be solved. To this end, we are pursuing a parallel research track investigating the use of projective invariants for image to image registration and for planar and 'nearly planar' scene reconstruction. The benefit of this approach is that the dependence on prior estimates of camera lens and pose parameters is minimised. One aspect of our work on invariants, automated image rectification (unwarping of oblique views) for matching coplanar structures, is described in these proceedings [Collins93].

2 Feature Extraction

In order to extend a partial site model from a set of images, it is necessary to determine where in each image the model appears. A model will normally be specified at a much higher level of geometric abstraction than the image intensity values. For this reason, feature extraction routines are first run on the images to pull out higher-level symbolic features of a type compatible with the features represented in the initial model. The type of geometric features that can potentially be extracted from the image includes straight line segments, line pencils, rectilinear line groupings, curves, corner points, regions of homogeneous intensity, and textured areas. Our current matching algorithm relies exclusively on straight line segments: edges of a wire-frame model are matched to straight lines extracted from the image.

2.1 Straight line extraction

We are applying two straight line segment extraction algorithms to the Radius model board imagery. The Burns algorithm [Burns86] begins by labeling pixels in the intensity plane according to coarsely quantised gradient orientation. A connect-components algorithm is then run to determine line-support regions, i.e. a set of pixels with an intensity surface that supports the presence of a straight line. Representative lines are extracted by intersecting a plane corresponding to the average intensity of a line-support region with a least-squares planar fit of the underlying inten-

sity surface of that region.

In contrast, the Boldt grouping algorithm [Boldt89] extracts local edges and then hierarchically groups them via geometric relations that were inspired by the Gestalt laws of perceptual organisation. The initial edges are the zero crossing points of the Laplacian of the intensity plane. In an iterative process, two edges are linked and replaced by a single longer edge if their end points are close and their orientation and contrast are similar, resulting in increasingly longer lines.

Our current implementation of the Burns algorithm runs much faster than Boldt, because it makes fewer local decisions at each stage of processing, and maintains fewer intermediate data structures. Indeed, a stripped down version of the Burns algorithm has been used as a fast line finder for robot navigation experiments [Fennema90]. Our current implementation of the Burns algorithm also works on larger image sizes. The one drawback is that oddly-shaped support regions caused by slow gradient intensity changes in the image can skew the orientation of the resulting lines. Current work is aimed at correcting this known problem. Figure 1 shows a portion of one of the model board images, while Figure 2 presents a set of lines extracted from this image by the Burns algorithm.

2.2 Vanishing point detection

Vanishing points can be an important source of information in urban and industrial scenes where buildings and roads are laid out in a rectangular grid. By grouping together pencils of lines that converge to a vanishing point, a higher level of geometric abstraction and data reduction is achieved. Under known camera lens parameters, vanishing points allow the computation of three-dimensional line and plane orientations from a single image, and thus allow the orientation of the camera with respect to the scene to be inferred (see Section 3.1). When the camera lens parameters are not known, they can be determined to a limited extent from vanishing point information [Wang91].

A practical algorithm for finding vanishing points from a set of line segments in an image must address two issues: how to cluster line segments going to a single vanishing point, and how to estimate an accurate vanishing point from a given line cluster. The former is handled elegantly using a Hough transform that maps line segments onto great circles in a histogram representing the surface of a unit sphere [Barnard83]. Potential vanishing points are detected as peaks in the histogram, corresponding to areas where several great circles intersect. While this approach excels at quickly clustering line segments into convergent groups, the final estimate of vanishing point location and variance should be based on the line segments themselves rather than the arbitrary bucket boundaries of a histogram data structure. We therefore use the Hough transform only as an initial clustering method and as an efficient

spatial access mechanism. The final vanishing point locations are computed using a statistical estimation technique that estimates each vanishing point location as the polar axis of an equatorial distribution on the unit sphere [Collins90]. The resulting polar axis points towards the location in the image where the converging lines intersect, and points parallel to the image plane when the underlying 2D line segments are parallel.

3 Model Matching

The second stage of the model extension process is model matching. Figure 3 shows a partial wireframe model constructed from the model board ground truth data. This model encompasses only those buildings where enough ground truth points were available to determine their shape and location. Note that some buildings are incompletely specified.

Given a partial 3D wireframe site model, and a set of extracted straight lines, the goal of model matching is to find the correspondence between model lines and data lines. To find this correspondence we are evaluating a novel model matching algorithm due to Beveridge [Beveridge90]. Based on the local search approach to combinatorial optimisation, this algorithm seeks the transformation that brings the projected model into subpixel alignment with the underlying image data.

The local search matching algorithm searches the discrete space of correspondence mappings between model and image features for one that minimises a match error function. The match error depends upon the relative placement implied by the correspondence, and the amount of coverage of the model by the data. More particularly, to compute the match error the model is placed in the scene so that the appearance of model features is most similar to the appearance of corresponding image features. The more similar the appearance the lower the match error. The mathematical transformation mapping model features to scene features is essentially a module of the system. Our current implementation handles the four parameter 2D similarity transform and the full 3D pose transform.

To find the optimal match, probabilistic local search relies upon a combination of iterative improvement and random sampling. Iterative improvement refers to a repeated generate-and-test procedure by which the algorithm moves from an initial match to one that is locally optimal via a sequence of incremental changes that continually reduce the match error. In an effort to find the global optimum, the algorithm is run multiple times, starting with different initial correspondences from the model to data line match space. Even if the probability of seeing the optimal match on a single trial is low, the probability of seeing the optimal match in a large number of trials started from uniformly random positions in the match space is high.

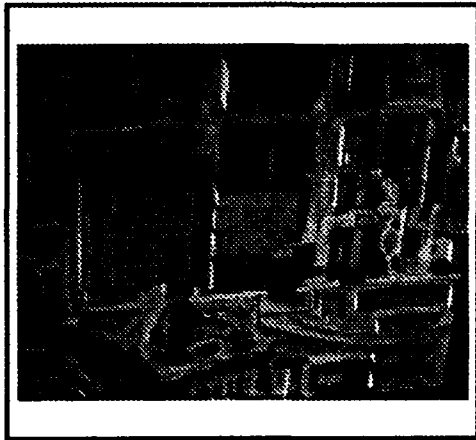


Figure 1: Model board image 8



Figure 2: Burns lines for image 8

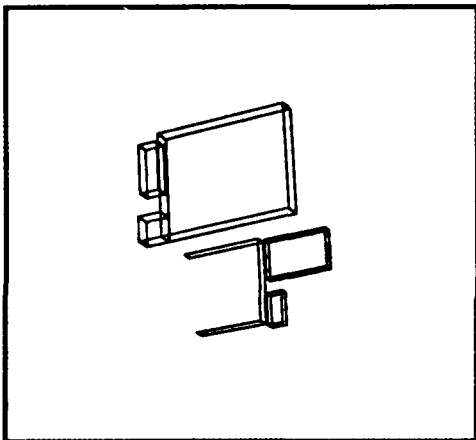


Figure 3: Partial wire-frame model

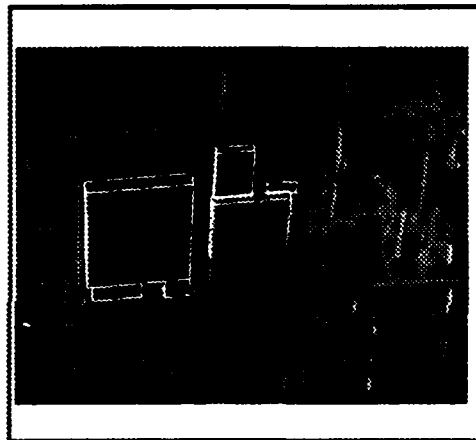


Figure 4: Initial model projection

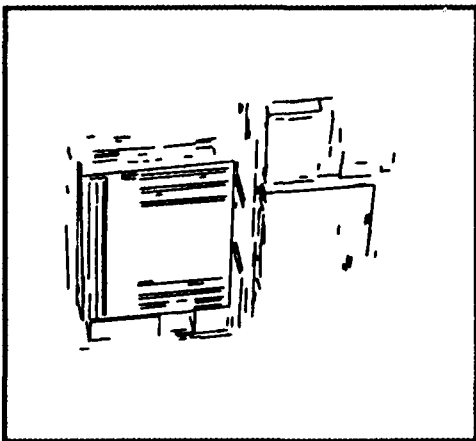


Figure 5: Candidate data correspondences

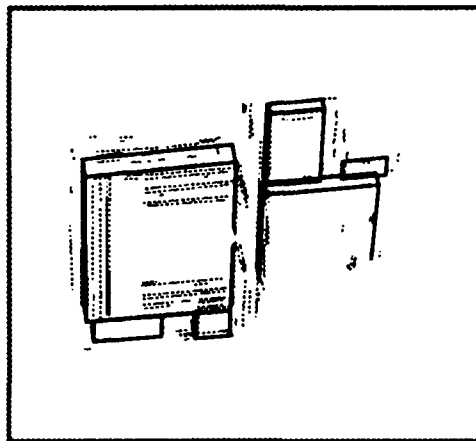


Figure 6: Best match of model to data

3.1 Initial camera model

Essentially all model-to-image correspondence problems involve solving both a discrete correspondence between model and image features along with an associated transformation mapping model features into the image. The two problems together constitute model matching: a match being a correspondence plus a transformation. The most general transformation typically considered involves full 3D pose: a rigid 3D model is rotated and translated relative to the camera and then projected into the image using a known camera model.

However, given good estimates of the lens and approximate pose parameters of the camera, the 3D model can be projected onto the image before matching begins, turning the problem into a search for the 2D transformation that best brings the *2D projected model lines* into correspondence with the data. This is the underlying motivation for the 2D similarity version of the model matcher. Finding the best 2D similarity transform between model and data is much faster than solving for full 3D pose. Because the method is fast, it is possible to run more trials in a given amount of time, thereby increasing the confidence in finding the best correspondence. Figure 4 displays a projection of the wire-frame site model onto our example model board image using initial estimates of the camera lens and pose parameters.

The camera parameters to be estimated for each image may be presented in the following 3 matrix form

$$\begin{array}{ccccc}
 3 \times 1 & & 3 \times 3 & & 3 \times 4 & 4 \times 1 \\
 k \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} & = & \begin{bmatrix} s_u & 0 & u_0 \\ 0 & s_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} R & t \end{bmatrix} & \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
 \end{array}$$

which maps the 3D coordinates x , y , and z of a model point into the 2D coordinates u and v of an image point. The transformation from model to image coordinates is broken into a 3×4 matrix of pose parameters and a 3×3 matrix of lens parameters. The pose parameter matrix is partitioned into a 3×3 orthonormal rotation matrix R and a 3×1 translation vector t . The lens parameters considered are s_u and s_v , the focal length in pixels along each of the image axes, and u_0 and v_0 , pixel coordinates of the principle point.

For our experiments, initial estimates of the lens and pose parameters for the Radius model board imagery were determined as follows. First, nominal values for the camera lens parameters were filled in from information supplied with the model board data (namely the focal length in mm and the dimensions of a pixel in mm), and by assuming the principle point to be in the numeric center of the image. The orientation of the camera with respect to the scene was determined by vanishing point analysis up to a four-fold ambiguity,

resolved by identifying the direction of true north in the image by hand. The distance of the camera from the ground was determined from the reported Ground Scale Distance (GSD); to date our experiments have only used the 18 inch GSD images. Finally, the intersection of the camera's line of sight with the ground plane was estimated manually (see Section 1.1 on the role of the imagery analyst).

3.2 Setting up the match space

Model matching performs a search through the space of possible model to data correspondences. This space is initially set up by deciding which data lines in the image are to be considered as candidate matches for each model line. Careful pruning of this space at the start is crucial to achieving tractable run times. Once again the problem is greatly simplified when good initial estimates of the camera transformation parameters are available. The better the initial estimates, the tighter the filters for picking out possible candidate data lines can be, both in terms of orientation, position in the image, and length.

Even though the metric used to score potential correspondences is purely geometric, photometric expectations such as the sign and magnitude of contrast across a line can be enforced in the final match by prefiltering for these properties in the initial candidate generation phase. Our tendency has been to underspecify rather than overspecify filter parameters, however, because once a correct line pairing has been excluded by overzealous filtering in the candidate generation stage, that correct pairing can never contribute to the match that is eventually found.

For the experiments we have run on the model board imagery, all lines located within 100 pixels and having an orientation within 10 degrees of a projected model line are selected as possible matching candidates. Figure 5 shows the complete set of candidate data lines considered in our example matching problem. Figure 6 displays the overlaid model after application of the 2D similarity transform associated with the best correspondence found.

4 Camera Resection

The result of model matching is a set of model to image feature correspondences between 3D wire-frame edges and 2D image lines. The next stage in the model extension process uses these correspondences to resect more accurate estimates of camera pose. These updated parameter estimates will be used to triangulate the positions of new scene features.

Kumar has developed optimisation techniques for finding 3D camera pose from point and line-based feature correspondences [Kumar92a, Kumar92b, Kumar92c]. His line-based constraints are similar to those devel-

oped by Liu, Huang and Faugeras [Liu88] from the observation that the 3D lines in the camera coordinate system must lie on the projection plane formed from the corresponding image line and the optical center. Using this fact, Liu et.al. separated the constraints for rotation from those of translation, leading to a solution in which rotation is solved for first, and then translation is obtained using the rotation results. Unfortunately, small errors in computing the rotation are amplified into large errors in translation.

Kumar's pose solution differs from that of Liu et.al. in two significant ways. First, rotation and translation are solved for simultaneously, which makes more effective use of the constraints and is more robust in the presence of noise. Second, the nonlinear least-squares optimisation algorithm used to solve for rotation and translation is adapted from Horn [Horn90]. Horn's method, based on the quaternion representation of rotations, provides much better convergence properties than solution methods based on Euler angles.

It is well known that least squares optimisation techniques are prone to errors when there are outliers in the data. Kumar developed a second suite of pose optimisation methods using robust statistics in order to minimise the effect of outliers. In these algorithms, the median of the error function is minimized, rather than the mean squared error. This approach is robust over data sets containing up to 50% outliers, at the expense of the increased computation needed to sample multiple subsets of data to find one devoid of outliers.

Based on a model of image noise and the assumption that the 3D model data is accurate, closed form expressions for the uncertainty in the pose refinement results (rotation and translation) have been derived. Kumar has shown analytically that the error in the output parameters is linearly related to the noise in the input data [Kumar92b]. He also studied the effect of errors in estimates of the image center and focal length on the resulting pose, showing that incorrect knowledge of the camera center does not significantly affect the computed 3D location of the sensor (although the computed rotation is affected), and that incorrect estimation of the camera focal length significantly affects only the z-component (depth) of the computed pose.

For images where the lens parameters are not available, or not known very accurately, the pose determination process could conceivably be extended to solve for both lens and pose parameters. The resulting highly nonlinear set of equations could best be solved if multiple images taken with the same camera were available, in which case a joint optimisation procedure could be used to determine the single set of lens parameters at the same time the pose parameters for each view were computed. We are investigating the feasibility of this approach for general applications.

5 Triangulation

Our approach to model extension began with the search for correspondences between a partial site model and geometric image features. Finding these correspondences and computing the camera pose relating the model coordinate system and the image coordinate system of each view has been discussed. Now, using the computed model to image transformations, correspondences of unmodeled features over the multiple views can be backprojected to locate new 3D model points and lines in the model coordinate system by triangulation.

Currently, only code for triangulation of point features is implemented. The estimation of new 3D points can be done in either batch or iterative sequential mode. Triangulation requires at least two frames and therefore the minimum batch size is two. Results from batch to batch can be integrated by the standard Kalman-filter covariance based updating equations.

Due to noise both in image measurements and camera pose estimates, image projection rays will not exactly intersect at a point. Kumar has developed a 3D pseudo-intersection method that minimises an error equation based on the same constraints that determine the pose [Kumar92b]. The criterion underlying this error equation is that the best estimate for any model point location is the point that minimises the least-squares distance between the predicted image location of the projected model point and its actual image location, taking into account covariances in the measured image positions and the computed pose. Two non-linear error equations are obtained for each scene point for each image frame, thus a minimum of two frames is needed to solve the system of equations. Techniques for the solution of nonlinear systems of equations generally require an initial estimate that is close to the true solution. The initial estimate in this case is chosen as the point that minimises the sum of squares of perpendicular distances to all the image projection rays, a point that is easily found by solving a linear system of equations. Using this initial guess, an iterative procedure is employed to solve the system of non-linear equations for each point. The iterative procedure is repeated until there is convergence. Usually only one iteration is sufficient for accurate results [Kumar92b]. A byproduct of this calculation is an approximate covariance matrix for the derived 3D model point position.

The method described above can also be used for model refinement. In this case initial model points have input covariances associated with them. The pseudo-intersection method is used to calculate a new estimate for each initial model point. The covariance matrices of a new estimate and an initial model point are used to fuse the two estimates and provide a new uncertainty matrix using the standard Kalman filtering equations.

Kumar assumes that the lens parameters are known, so only uncertainty in the pose parameter estimates is considered when computing the error in a triangulated point position. We are extending this approach to handle uncertainty in the camera lens parameters as well. We are also extending the triangulation equations to work for lines as well as points.

6 Summary

A system for automated site model matching and extension is under development at the University of Massachusetts. A sketch of the final system has been presented, based on algorithms that have already been developed for performing feature extraction, model matching, pose determination, and triangulation. These algorithms are currently being tested on the Radius model board imagery. Given the excellent model extension results we have obtained in past robot navigation applications, we are optimistic about the utility of these routines in the aerial photogrammetry domain.

References

- [Barnard83]
S.T.Barnard, "Interpreting Perspective Images," *AI Journal*, Vol. 21(4), 1983, pp. 435-462.
- [Beveridge90]
J.R.Beveridge, R.Weiss, and E.M.Riseman, "Combinatorial Optimisation Applied to Variable Scale 2D Model Matching," *Proc. ICPR*, Atlantic City, June 1990, pp. 18-23.
- [Beveridge92a]
J.R.Beveridge and E.M.Riseman, "Can Too Much Perspective Spoil the View? A Case Study in 2D Affine Versus 3D Perspective Model Matching," *Proc. Darpa IUW*, San Diego, 1992, pp. 655-663.
- [Beveridge92b]
J.R.Beveridge and E.M.Riseman, "Hybrid Weak-Perspective and Full-Perspective Matching," *Proc. CVPR*, Champaign, IL, 1992, pp. 432-438.
- [Burns86]
J.B.Burns, A.R.Hanson, and E.M.Riseman, "Extracting Straight Lines," *IEEE Trans. PAMI*, Vol. 8(4), July 1986, pp. 425-455.
- [Boldt89]
M.Boldt, R.Weiss and E.Riseman, "Token-Based Extraction of Straight Lines," *IEEE Trans. SMC*, Vol. 19(6), 1989, pp. 1581-1594.
- [Collins90]
R.T.Collins and R.S.Weiss, "Vanishing Point Calculation as a Statistical Inference on the Unit Sphere," *Proc. ICCV*, Osaka, 1990, pp. 400-403.
- [Collins93]
R.T.Collins and J.R.Beveridge, "Matching Perspective Views of Coplanar Structures using Projective Unwarping and Similarity Matching," *Proc. Darpa IUW*, Washington, DC, 1993.
- [Fennema90]
C.Fennema, A.Hanson, E.Riseman, J.R.Beveridge and R.Kumar, "Model-Directed Mobile Robot Navigation," *IEEE Trans. SMC*, Vol. 20(6), 1990, pp. 1352-1369.
- [Horn90]
B.K.P.Horn, "Relative Orientation," *Int. Journal of Computer Vision*, Vol. 4, 1990, pp. 59-78.
- [Kumar92a]
R.Kumar and A.R.Hanson, "Application of Pose Determination Techniques to Model Extension and Refinement," *Proc. Darpa IUW*, San Diego, CA, January 1992, pp. 727-744.
- [Kumar92b]
R.Kumar, *Model Dependent Inference of 3D Information from a Sequence of 2D Images*, Ph.D. Thesis, Department of Computer Science, University of Massachusetts, February 1992. Also published as CS tech report TR92-04.
- [Kumar92c]
R.Kumar, H.Sawhney, and A.Hanson, "3D Model Acquisition from Monocular Image Sequences," *Proc. CVPR*, Champaign, IL, 1992, pp. 209-215.
- [Liu88]
Y.Liu, T.S.Huang and O.Faugeras, "Determination of Camera Location from 2D to 3D Line and Point Correspondences," *Proc. CVPR*, Ann Arbor, 1988, pp. 82-88.
- [Wang91]
L.L.Wang and W.H.Tsai, "Camera Calibration by Vanishing Lines for 3D Computer Vision," *IEEE Trans. PAMI*, Vol. 13(4), April 1991, pp. 370-376.

Site-Model-Based Change Detection and Image Registration

R. Chellappa, Q. Zheng, L.S. Davis, D. DeMenthon, A. Rosenfeld

Computer Vision Laboratory, Center for Automation Research,

University of Maryland, College Park, MD 20742-3275

Abstract

The University of Maryland (with TASC as a subcontractor) is one of the group of institutions doing research on aerial image understanding in support of the RADIUS program. The emphasis of our research is on knowledge-based change detection (CD) using site models and the domain expertise of image analysts (IAs). Change detection involves classifying changes in the imagery as being due to site updates or activity, or as irrelevant changes due to illumination differences, seasonal variations, etc. The IA's expertise is crucial in identifying relevant changes, which depend on the site and the intelligence agenda. Our focus is on ways in which image understanding (IU) techniques can aid the IA in performing CD. We are designing a system that allows the IA to specify what are to be considered as relevant changes, and to select appropriate IU algorithms for detecting these changes.

Before CD can be attempted, the acquired images have to be registered to the site model. We are developing efficient constrained search mechanisms for image-to-site model registration, using techniques based on non-monotonic reasoning (Assumption-based Truth Maintenance Systems (ATMSs) and their variants). We are also using such techniques to facilitate interactive IA guidance for CD and site model updating.

1 Introduction

The process of locating and identifying significant changes or new activities, known as change detection (CD), is one of the most important imagery exploitation tasks [1]. Previous research on CD has emphasized the development of general-purpose methods that can be employed to screen a wide variety of imagery and determine, without access to any site-specific model information, whether any significant changes or events have occurred between the times of acquisition of the imagery. These methods have been found to be unreliable because a) CD techniques based on more or less sophisticated differencing of images (possibly after attempted corrections

for viewpoint and illumination differences) are extremely sensitive to errors in registration and in the photometric models (e.g. reflectance, illumination) that are used; b) too many inconsequential changes occur in any natural environment. Even if general-purpose methods could be developed for screening out all changes due to variations in viewpoint, sensor and illumination, there would still be many differences between the images whose significance could only be determined by an image analyst (IA) using comprehensive site knowledge and the relevant intelligence agenda. Thus the goal of relieving the IA of the burden of screening large subsets of acquired imagery is unlikely to be achieved using such general-purpose methods.

We plan, instead, to develop a model-based vision system for CD incorporating image understanding (IU) techniques whose primitives are specific to a particular site type, and that can be employed by the IA to direct the IU system to conduct spatially constrained analyses whose outcomes may be indicative of occurrences of changes that have intelligence significance. The system will be site-model based, employ a heavily visual man/machine interface, and will be based on three classes of primitives: object primitives, which correspond to the specific objects that occur in a particular site model and to the generic object classes supported by the IU system; spatial primitives, for the construction of search locales and the specification of constraints on the search for object types within locales; and temporal primitives, which can constrain or parameterize the analysis by factors such as time of day, day of week, time of year, etc. The system will assist the IA by highlighting areas on an image where there are relevant activities, new or upgraded facilities.

As reported in [1], IAs have identified two ways in which IU can be useful in CD: the "quick-look" (QL) and "final-look" (FL) modes. In the QL mode, small areas where any change would be considered significant are declared a priori, and when the system is presented with a series of images, only those that satisfy the conditions in the QL profile are marked. In the FL mode, a set of less important areas to be examined for change is specified. These areas are less important, but the IA wants to examine them to ensure complete coverage of the site. As the IA gains experience, both the QL and FL profiles can be modified. The CD system that we

plan to build could include these two options.

The site models considered in the current phase of RADIUS encode only the spatial relationships between fixed objects of interest in a site, such as buildings, roads, etc. An important issue in training new analysts or reviewing infrequently analyzed sites is the coding of the temporal relationships which describe changes in the site such as movements of vehicles under normal or abnormal circumstances—i.e., a site activity model. The CD system described above will be a valuable step toward the development of a site activity modeling capability.

Generally the first step in a CD task is the registration of an image to an existing site model. Depending on the CD task, using the existing site model and camera parameters, regions of interest in the given image will first be outlined. Subsequently, objects such as buildings and vehicles that are characteristically present in the site can be extracted and analyzed for CD purposes. Such object extraction algorithms cannot be purely bottom-up. For example, in extracting buildings [2], heuristics based on the expected shapes of roofs (site-specific information) are very useful for completing any partial roof hypotheses that result from imperfect bottom-up processing. Likewise, shadow analysis is very useful for obtaining height information [3, 4], or allowing the IU system to explain why some building features that are in the field of view cannot be identified in the image. Site models can also be very useful for providing geometric and photometric constraints that reduce matching ambiguities.

In addition to image-to-site model registration, we are also interested in image-to-image registration where two images acquired from possibly severe off nadir viewing conditions need to be registered. Image-to-image registration is useful for building site models, and for performing the subtask of transforming a given image to a "favored orientation" [1]. The images to be analyzed as part of the RADIUS-related research program are high-resolution images of complicated sites. In many of the currently used image registration algorithms, tie points need to be manually selected. This can be a laborious task. Automatic registration of the two images is desirable. Given the variability of viewing directions, illumination conditions and resolution, the features used for matching may be poorly localized or occluded. Automatic image-to-image registration will be accomplished using appropriate cues from site models and camera models. Although the use of site models for image registration and CD has attractive features, there are some problems associated with this approach. Site models are usually not complete; only a few images may have been used in building a site model. Inferences based on incomplete site models may be erroneous or incomplete. The matching algorithms used in image registration should be able to deal with uncertainties. Currently used matching algorithms based on relaxation, interpretation trees [5] and constraint satisfaction networks [6] lead to inefficient, repetitive and uncontrolled search for matches. We propose to use search methods based on non-monotonic reasoning—in particular, Assumption-based Truth Maintenance Sys-

tems (ATMSs) [7, 8]. ATMSs achieve efficiency by keeping track of unsuccessful or undesirable search paths. Conditions that lead to unfruitful search are declared and stored as *nogoods*. These *nogoods* can be declared a priori or during analysis, based on the IA's expertise in terms of undesirable matchings or in terms of inconsistencies of features and their groupings as predicted by the rendered scenes or the site models themselves.

ATMSs can also be used for updating a site model once the CD module and the IA identify the changes of consequence. This process involves additions and deletions of assertions or hypotheses that exist as part of the site model. The subproblems involved are generating symbolic descriptions of the image regions where changes have occurred and incorporating these descriptions into the model.

It is evident that the IA will perform a crucial role in directing, manipulating and correcting the results of IU algorithms. An important part of our approach is the inclusion of early feedback, by users familiar with the final application, as to the usability of the algorithms developed under this program. Although formal usability test sessions are not envisioned, subjects will be asked to perform routine and specialized tasks for evaluation. These evaluations will provide valuable information with respect to the likely models and levels of interaction to be expected from IA's, the clarity and intuitive understandability of the IU algorithms, and whether the typical IA is able to tailor the responses of the algorithm to his/her needs.

The interactions of the various modules described above are illustrated in Figure 1.

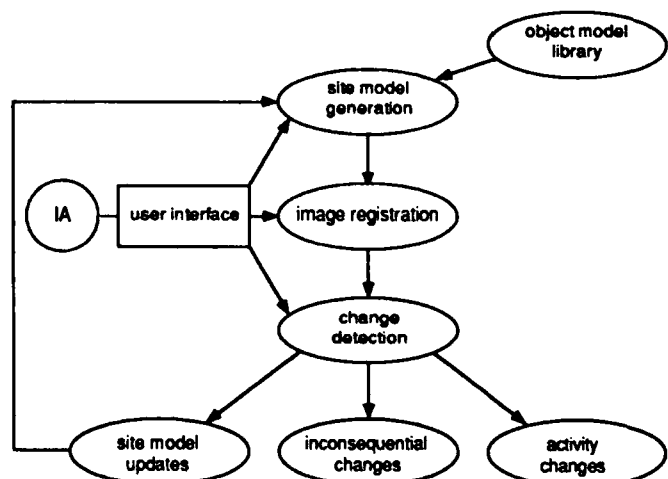


Figure 1: Schematic representation of the change detection process.

2 Research Areas

2.1 Site-Specific System for Change Detection

We are developing a system whose primitives are specific to a particular site type. This system will be employed by the IA to direct the IU system to determine the oc-

currence of changes that have intelligence significance. The system will have the following characteristics:

1. It will be **site-model based**—i.e., the specification of changes of interest will be in terms of object types in the site model, type-specific changes, and three-dimensional spatial relations that can be used to direct and constrain the IU algorithms. This is critical because these changes must be detected in future imagery whose source characteristics (view-points, spatial resolution) will not be known in advance.
2. The man/machine interface will be **visual**. The IA will construct a change specification by graphical manipulation of the current elements of the site model and of generic class models supported by the IU system (e.g. vehicle classes, road models), and insertion of visual tokens into the site model that will provide to the IU system information concerning where to search, what to search for, and the specificity of the search.
3. The system design will be based on three classes of primitives: **object primitives**, which correspond to the specific objects that occur in a particular type of site model and the generic object classes supported by the IU system; **spatial primitives** for the construction of search locales and the specification of constraints concerning the search for object types within locales; and **temporal primitives** which might constrain or parameterize the analysis by factors such as time of day, day of week, time of year, etc.

We illustrate these ideas with two example tasks:

1. **Counting objects within locales.** Many types of changes involve counting instances of objects within locales and comparing these counts either against an absolute standard or with their values at previous times. For example, the analyst might wish to monitor the number of vehicles in a parking area and be informed if their number is ever above a given threshold. The specification of this change would involve:
 - (a) Indicating the locale of interest by either pointing to it in a visualization of the site model (if it is an explicit component of the site model) or using a graphical tool to indicate its position and extent. For example, while a parking lot may be an element of a site model, the analyst may only be interested in increases in vehicles in the part of the lot near some building of interest. In such a case, graphical tools can be used to construct an arbitrary locale for search.
 - (b) Choosing one or more vehicle model types from a menu of object classes.
 - (c) Choosing a "count" option from an analysis menu, and specifying the criteria for a significant change.
 - (d) Specifying temporal constraints on when to conduct the analysis. For example, it may be of interest to count vehicles only on weekdays.

2. **Activity modeling.** Detecting changes in a complicated environment often involves the fusion of multiple change cues. The individual changes may not be significant, but their simultaneous occurrence may be very significant. As an example, consider the monitoring of airfield activity. The arrival of a significant number of new aircraft at an airfield is not unusual in support of an ongoing training or exercise activity. Increased activity at the weapons storage facility associated with the airfield is also not uncommon in support of resupply and training activities. However, the simultaneous occurrence of the arrival of a large number of aircraft not normally seen, together with increased activity at the weapons storage area, might indicate preparations for hostile actions. The interface will be designed to incorporate specification of interactions between cues, defined in terms of object, spatial, or temporal primitives.

2.2 Registration Algorithms

We are investigating two types of registration processes, image-to-site model registration and image-to-image registration. Prior to any CD task, the newly acquired image needs to be registered to the existing site model. Depending on the particular CD task, e.g., if building or vehicle related activity is being monitored, we can use the site model and viewing direction of the new image to identify regions in the image that need further study. We can subsequently invoke the necessary IU algorithms related to building detection, vehicle location and counting (and road extraction, if construction of roads is monitored). For tasks such as these, there are two very important model-based IU techniques:

Registering images to site models. Consider, for example, the problem of identifying the region in an aerial image corresponding to a given parking lot. While estimates of sensor and platform parameters are known, it is not sufficient to simply project the parking lot boundaries onto the image plane using these parameters, since these parameters are subject to errors. Furthermore, determining which parts of the parking lot are visible in the image (since parts of the parking lot can be occluded by other elements of the site model) and the illumination conditions in the visible part of the parking lot (parts of which may be in shadow depending on sun angle and site model geometry) are critical to subsequently making a correct decision as to whether there is a significant difference between the numbers of observed and expected vehicles in the parking lot. In fact, the feasibility of performing a CD task depends on the IU system correctly modeling the relationship between a given image and the site model (for example, if we were interested in whether a large number of vehicles are parked near a certain building, it could be important to determine if that part of the parking lot is, in fact, visible in the image). As described in the next section, ATMS-based techniques can also be used for registering images to site models.

Model-based object recognition. Many CD tasks involve the identification of objects in constrained parts of the image, along with an analysis of their density and spatial distribution. If the image has sufficient spatial resolution, and CAD models are available for the objects of interest, then IU techniques such as alignment [9], and geometric hashing [10] are appropriate. Given that good estimates of the height and orientation of the sensor with respect to the ground plane are known a priori (and can be improved by the site model to image registration process), a method such as the one developed at Maryland by Silberberg et al. [11] can be employed; this method has the minimal combinatorial complexity of any 3-D object recognition algorithm, requiring that only a single image feature be matched against a single model feature to determine the image to model transformation. In situations where the spatial resolution is not sufficient, or where geometric models are not available, one can employ general-purpose model-based image segmentation algorithms such as Programmed Picture Logic (PPL) [12], which can be extended to "learn" the descriptions of objects of interest (through a few visual examples), and can then identify remaining instances viewed under similar conditions of viewpoint and illumination.

In addition to image-to-site model registration, which will be directly useful for CD, we are also developing a general-purpose image-to-image registration algorithm. Such an algorithm will be useful for building site models and for orienting an image in a "favored position". The traditional stereo paradigm [13] for inferring 3-D structure is not applicable to images acquired from severe off-nadir viewing directions. Our goal is to develop a completely automatic registration algorithm using site models and any auxiliary information such as camera parameters. Site models will be very useful for registering two severely off-nadir images, as we can predict the contrasts of features in both images, occlusions of features and shadow regions.

An integral component of site model based registration and change detection is the availability of site models. We are working on site model construction and updating on an ongoing basis. The solution to site model construction assumes that several overlapping coverage images are available. We will initially construct a site model using the RCDE site model rendering system. Image-to-model registration algorithms will be used to register each image to the site model. When two or more images confirm the same hypotheses about the underlying object, the initial assertions about the object will be replaced by image-derived assertions. This will be done in an incremental fashion. During the early stages, the errors due to incomplete specification of site models may be handled by allowing more tolerance in the predicted positions of features and their computed attributes. As more images become available, the representation error will decrease.

2.3 Truth Maintenance

Algorithms for CD and image registration need to perform model based search. Any search method used for these tasks should have the following characteristics:

- 1) It should be able to handle uncertainties in the locations and attributes of features due to errors in feature extraction, incomplete site models, etc.
- 2) It should be efficient. Since the high resolution RADIIUS images may produce large numbers of acceptable features, the search space may be prohibitively large. Efficiency is achieved by avoiding futile backtracking and eliminating rediscovery of inferences.
- 3) It should be permitted differential diagnosis so that solutions can be directly compared with one another at different points in the search space and the "best" interpretation chosen. (Note that more than one reasonable solution may be found in real life situations.)
- 4) It should be able to make use of geometric and photometric constraints derived from site models.
- 5) It should be able to accept guidance from the user.

The second requirement eliminates exhaustive search and depth-first search, also known as chronological backtracking. In chronological backtracking [14], when the search backtracks because it encounters contradictions, it forgets any inferences made in that portion of the search space. These inferences will be rediscovered at other places. Also, the underlying reasons for the contradictions are not stored, so that these contradictions may be encountered again and again. As a solution to this "forgetting" problem, one may consider dependency-directed backtracking [15]. In this scheme, dependency records are maintained for each inference. These records link each inference to its antecedents. When a contradiction is encountered, the dependency records are used to backtrack to the most recent selection which actually contributed to the contradiction, instead of resorting to chronological backtracking. This avoids futile backtracking. Also, the dependency records are bidirectional and are used to reinstate previously derived information in different portions of the search space, thus avoiding rediscovery of inferences. When a contradiction is encountered the underlying assumptions that actually led to the conflict are stored for future reference as *nogoods*. These *nogoods* are used to prevent rediscovery of contradictions, and hence improve efficiency.

Truth maintenance systems (TMSs) of various kinds employ dependency-directed backtracking. A justification based truth maintenance system (JTMS) [7] works with nodes each of which corresponds to a problem solver datum. Associated with each node is a justification which summarizes how the node originated. Truth maintenance is a procedure that decides which problem solver is to be believed (in) and which disbelieved (out). It uses dependency-directed backtracking for this purpose.

In our registration algorithms we plan to use an ATMS [8, 16, 17] for search. An ATMS works by manipulating assumption sets, which are primitive data from which all other data are derived. It works on the principle that assumption sets can be manipulated much more conveniently than the data sets they represent. It simplifies truth maintenance and eliminates backtracking. When using an ATMS, all contexts (points in the search space) are simultaneously visible to the problem solver. This

permits differential diagnosis. An ATMS will find all the solutions that pure enumeration would. At the same time, it improves the efficiency of the problem solver without sacrificing coherence and completeness.

An important advantage of using an ATMS is its ability to integrate IA-derived and site-model-derived information. The *nogoods* mentioned above can be set by the IA. For example, if the sunlight is diffuse, so that crisp shadow information is unlikely to be available, the use of shadows can be declared as a *nogood* condition. All inferences that use shadow information will then be avoided and the final result will be one which did not use any information about shadows. As an example of site-model-derived constraints, the relative positions of features in the two images to be registered must be consistent with those predicted by the site model and any violations can be declared as *nogoods*. Topological constraints derived from site models can also be used to define *nogoods*.

The ATMS will also be useful for handling the belief revisions involved in site model updating. In an ATMS uncertainty is dealt with by making *assumptions* (*hypotheses*) whose beliefs can be later revised as new evidence accumulates. Another advantage in using an ATMS is that it provides mechanisms for the automatic construction of the inference network. A very simple version of a JTMS was used in 3D MOSAIC [18] for incremental reconstruction of site models in a simple domain. Since an ATMS is much more versatile than a JTMS, we will be able to handle the more complicated site models that arise in RADIUS applications.

2.4 Usability Analysis

One of the key contributions of TASC to our project will be to assist us in developing IU capabilities which are amenable to use by IAs. This assistance will be critical in two main areas: support of the ATMS-based registration and site-model updating algorithm development, and assistance in developing the specification interface for goal-directed CD.

The expectation is that ATMS-based techniques will be developed in such a way that they can be integrated into an interactive environment. Such interaction must satisfy the following constraints:

- 1) The algorithms developed must provide rapid feedback to the user about their current state and the quality of the product being generated.
- 2) Any global algorithm parameters required to be set by the IA must have an intuitive interpretation. The appropriate values of these parameters should depend in a direct manner on factors easily accessible to the IA, for example, sun angle and general image quality.
- 3) The effect of this parameterization on algorithm behavior should also be intuitive.
- 4) An interactive capability should be available to interject additional IA guidance as the algorithm progresses.
- 5) The ability of the algorithms to backtrack is crucial. The IA should be able to assert new information or

controvert results obtained by the IU system.

A similar constructive role will be played by the TASC team in supporting the development of the specification interface for goal-directed CD. The technology is not ripe for automatic translation of IA queries in natural language to define the set of IU tasks that need to be performed. Even preliminary efforts and successes in bringing IU algorithms a step closer to the eventual users in the intelligence community will be worthwhile. There are several issues to be addressed from an IA point of view:

- 1) The interface must be powerful enough to communicate a broad variety of objectives.
- 2) The primitives must relate directly to the analysis paradigm employed by the analyst.
- 3) The analyst must be confident that the specification interface has encoded all the constraints that he/she has imposed.

An IA must have significant confidence in the system in order to be willing to use it. It is impossible to verify that all changes of relevance have been detected without human review.

3 Accomplishments to Date

During our first four months under contract (as of the time of the writing of this paper) we have made considerable progress on several fronts: (1) We have developed a novel image-to-image registration algorithm that can automatically register two off-nadir images. When additional information about camera parameters is available, the algorithm becomes considerably simpler. More details about the algorithm, and experimental results obtained on model board and real images, are given in the remainder of this section. (2) We have tested the line detector developed by Venkateswar and Chellappa [19] on model board imagery. The line outputs will be useful for building detection [2] algorithms.

In the following subsections, we first discuss the transform between images taken from two off-nadir orientations. We then present our image registration algorithm, including details such as feature point extraction and match verification. Experimental results on model board and real aerial images provided by the sponsor are presented.

3.1 3-D Transform

Some notation In the following discussion we use the notation given below:

$$T_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix} \quad (1)$$

$$T_z(\theta) = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$\epsilon = \frac{c}{f} \quad (3)$$

where ϵ is the resolution when digitizing an image, and f is the focal length of the camera.

Off-nadir to nadir transform We begin by considering the transformation from off-nadir camera coordinates $x_c y_c z_c$ to the world coordinate system $x_w y_w z_w$ with $x_w y_w$ plane parallel to the ground plane. The camera coordinates are characterized by the triplet (α, β, γ) where α is the angle from the north direction to the positive direction of the camera rotation axis, β is the angle from z_w to z_c , and γ is the angle from the x axis to the north direction.¹ Figure 2 shows the definitions of α , β and γ .

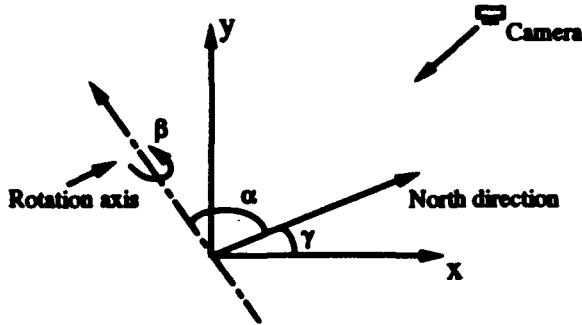


Figure 2: Camera rotation angles

Let the triplet (α, β, γ) be $(\alpha_w, \beta_w, \gamma_w)$ when measured in world coordinates and $(\alpha_c, \beta_c, \gamma_c)$ when measured in camera coordinates. For real applications, normally α and β are available in the world coordinate system while γ is defined with respect to the camera (image) coordinate system. The relationship between the two triplets is

$$\begin{aligned}\beta_c &= \beta_w = \beta \\ \alpha_c &= \arctan \frac{\sin \alpha_w \cos \beta}{\cos \alpha_w} \\ \alpha_c + \gamma_c &= \alpha_w + \gamma_w\end{aligned}$$

or

$$\beta_c = \beta_w = \beta \quad (4)$$

$$\alpha_c = \arctan(\tan \alpha_w \cos \beta) \quad (5)$$

$$\gamma_c = \alpha_w + \gamma_w - \arctan(\tan \alpha_w \cos \beta) \quad (6)$$

The 3-D transform from the world coordinates to the camera coordinates can be written in terms of rotations with respect to the x and z axes defined in (1) and (2) as

$$\begin{aligned}\vec{V}_c &= T_z(-\alpha_c - \gamma_c + \alpha_w + \gamma_w) T_x(-\alpha_w - \gamma_w) \cdot \\ &\quad T_x(\beta) T_z(\alpha_w + \gamma_w) \vec{V}_w + \vec{V}_{wc} \\ &= T_z(-\alpha_c - \gamma_c) T_x(\beta) T_z(\alpha_w + \gamma_w) \vec{V}_w + \vec{V}_{wc} \quad (7)\end{aligned}$$

where \vec{V}_{wc} accounts for the 3-D translation.

The transformation from an off-nadir system to the world system is

$$\vec{V}_w = T_z(-\alpha_w - \gamma_w) T_x(-\beta) T_z(\alpha_c + \gamma_c) \vec{V}_c + \vec{V}_{cw} \quad (8)$$

where

$$\vec{V}_{cw} = -T_z(-\alpha_w - \gamma_w) T_x(-\beta) T_z(\alpha_c + \gamma_c) \vec{V}_{wc}$$

¹The reason for using triplet (α, β, γ) to represent the camera orientation is that in real applications, the north direction is used as the reference to define the camera rotation axis.

Off-nadir to off-nadir transform Consider two off-nadir cameras characterized by triplets $(\alpha_{c1}, \beta_1, \gamma_{c1})$ and $(\alpha_{c2}, \beta_2, \gamma_{c2})$ respectively. From (7) and (8), the transformation from one off-nadir system to the other is

$$\begin{aligned}\vec{V}_{c2} &= T_z(-\alpha_{c2} - \gamma_{c2}) T_x(\beta_2) T_z(\alpha_{w2} + \gamma_{w2}) \cdot \\ &\quad T_z(\gamma_{w1} - \gamma_{w2}) [T_z(-\alpha_{w1} - \gamma_{w1}) T_x(-\beta_1) \cdot \\ &\quad T_z(\alpha_{c1} + \gamma_{c1}) \vec{V}_{c1} + \vec{V}_{cw1}] + \vec{V}_{wc2} \\ &= T_z(-\alpha_{c2} - \gamma_{c2}) T_x(\beta_2) T_z(\alpha_{w2} - \alpha_{w1}) \cdot \\ &\quad T_x(-\beta_1) T_z(\alpha_{c1} + \gamma_{c1}) \vec{V}_{c1} + \vec{V}_{12} \\ &= T_z(-\varphi_2) T_x(\beta_2) T_z(\theta) T_x(-\beta_1) T_z(\varphi_1) \vec{V}_{c1} + \vec{V}_{12} \\ &= R \vec{V}_{c1} + \vec{V}_{12} \quad (9)\end{aligned}$$

where

$$\varphi_2 = \alpha_{c2} + \gamma_{c2} = \alpha_{w2} + \gamma_{w2} \quad (10)$$

$$\varphi_1 = \alpha_{c1} + \gamma_{c1} = \alpha_{w1} + \gamma_{w1} \quad (11)$$

$$\theta = \alpha_{w2} - \alpha_{w1} \quad (12)$$

$$\vec{V}_{12} = \begin{pmatrix} r_{14} \\ r_{24} \\ r_{34} \end{pmatrix}$$

$$= T_z(-\varphi_2) T_x(\beta_2) T_z(\alpha_{w2} + \gamma_{w1}) \vec{V}_{cw1} + \vec{V}_{wc2} \quad (13)$$

$$\begin{aligned}R &= T_z(-\varphi_2) T_x(\beta_2) T_z(\theta) T_x(-\beta_1) T_z(\varphi_1) \\ &= \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \quad (14)\end{aligned}$$

$$\begin{aligned}r_{11} &= \cos \theta \cos \varphi_1 \cos \varphi_2 \\ &\quad - \sin \theta \sin \varphi_1 \cos \varphi_2 \cos \beta_1 \\ &\quad + \sin \theta \cos \varphi_1 \sin \varphi_2 \sin \beta_2 \\ &\quad + \cos \theta \sin \varphi_1 \sin \varphi_2 \cos \beta_1 \cos \beta_2 \\ &\quad + \sin \varphi_1 \sin \varphi_2 \sin \beta_1 \sin \beta_2 \quad (15)\end{aligned}$$

$$\begin{aligned}r_{12} &= \cos \theta \sin \varphi_1 \cos \varphi_2 \\ &\quad + \sin \theta \cos \varphi_1 \cos \varphi_2 \cos \beta_1 \\ &\quad + \sin \theta \sin \varphi_1 \sin \varphi_2 \cos \beta_2 \\ &\quad - \cos \theta \cos \varphi_1 \sin \varphi_2 \cos \beta_1 \cos \beta_2 \\ &\quad - \cos \varphi_1 \sin \varphi_2 \sin \beta_1 \sin \beta_2 \quad (16)\end{aligned}$$

$$\begin{aligned}r_{13} &= -\sin \theta \cos \varphi_2 \sin \beta_1 \\ &\quad + \cos \theta \sin \varphi_2 \sin \beta_1 \cos \beta_2 \\ &\quad - \sin \varphi_2 \cos \beta_1 \sin \beta_2 \quad (17)\end{aligned}$$

$$\begin{aligned}r_{21} &= \cos \theta \cos \varphi_1 \sin \varphi_2 \\ &\quad - \sin \theta \sin \varphi_1 \sin \varphi_2 \cos \beta_1 \\ &\quad - \sin \theta \cos \varphi_1 \cos \varphi_2 \cos \beta_2 \\ &\quad - \cos \theta \sin \varphi_1 \cos \varphi_2 \cos \beta_1 \cos \beta_2 \\ &\quad - \sin \varphi_1 \cos \varphi_2 \sin \beta_1 \sin \beta_2 \quad (18)\end{aligned}$$

$$r_{22} = \cos \theta \sin \varphi_1 \sin \varphi_2$$

$$\begin{aligned}
& + \sin \theta \cos \varphi_1 \sin \varphi_2 \cos \beta_1 \\
& - \sin \theta \sin \varphi_1 \cos \varphi_2 \cos \beta_2 \\
& + \cos \theta \cos \varphi_1 \cos \varphi_2 \cos \beta_1 \cos \beta_2 \\
& + \cos \varphi_1 \cos \varphi_2 \sin \beta_1 \sin \beta_2
\end{aligned} \quad (19)$$

$$\begin{aligned}
r_{23} = & - \sin \theta \sin \varphi_2 \sin \beta_1 \\
& - \cos \theta \cos \varphi_2 \sin \beta_1 \cos \beta_2 \\
& + \cos \varphi_2 \cos \beta_1 \sin \beta_2
\end{aligned} \quad (20)$$

$$\begin{aligned}
r_{31} = & \sin \theta \cos \varphi_1 \sin \beta_2 \\
& + \cos \theta \sin \varphi_1 \cos \beta_1 \sin \beta_2 \\
& - \sin \varphi_1 \sin \beta_1 \cos \beta_2
\end{aligned} \quad (21)$$

$$\begin{aligned}
r_{32} = & \sin \theta \sin \varphi_1 \sin \beta_2 \\
& - \cos \theta \cos \varphi_1 \cos \beta_1 \sin \beta_2 \\
& + \cos \varphi_1 \sin \beta_1 \cos \beta_2
\end{aligned} \quad (22)$$

$$\begin{aligned}
r_{33} = & \cos \theta \sin \beta_1 \sin \beta_2 \\
& + \cos \beta_1 \cos \beta_2
\end{aligned} \quad (23)$$

3.2 Image Transform

For nadir images, the ground plane is parallel to the image plane, leading to a simple relationship between the image plane coordinates and the 3-D x and y coordinates. When dealing with off-nadir images, there is a slant component in the depth, depending on the distance between the ground point and the camera rotation axis. In our triplet (α, β, γ) camera rotation model, the equations for the camera rotation axis are

$$\begin{aligned}
x_{c1} \sin \varphi_1 - y_{c1} \cos \varphi_1 &= 0 \\
x_{c1} &= z_o
\end{aligned}$$

Hence the distance from a point (x_{c1}, y_{c1}, z_{c1}) to the camera rotation axis in the $o_c x_c y_c$ plane is

$$d = x_{c1} \sin \varphi_1 - y_{c1} \cos \varphi_1 \quad (24)$$

and the depth z_{c1} can be determined by

$$\begin{aligned}
z_{c1} &= z_o + (x_{c1} \sin \varphi_1 - y_{c1} \cos \varphi_1) \tan \beta_1 \\
&= z_o + Ax_{c1} + By_{c1}
\end{aligned} \quad (25)$$

where

$$\begin{aligned}
A &= \sin \varphi_1 \tan \beta_1 \\
B &= -\cos \varphi_1 \tan \beta_1
\end{aligned}$$

z_o is the distance from the camera to the center of the scene.

Let (X, Y) be the image plane coordinates (indexes); we then have

$$\begin{aligned}
\frac{X\varepsilon}{f} = X\varepsilon &= \frac{x}{z} \\
\frac{Y\varepsilon}{f} = Y\varepsilon &= \frac{y}{z}
\end{aligned}$$

The depth in the first camera coordinate system can be represented in terms of the image plane coordinates

as

$$\begin{aligned}
z_{c1} &= Ax_{c1} + By_{c1} + z_o \\
&= AX_1\varepsilon_1 + BY_1\varepsilon_1 + z_o \\
&= \frac{z_o}{1 - AX_1\varepsilon_1 - BY_1\varepsilon_1}
\end{aligned} \quad (26)$$

The relationship between the two image planes is

$$\begin{aligned}
X_2\varepsilon_2 &= \frac{x_{c2}}{z_{c2}} \\
&= \frac{r_{11}x_{c1} + r_{12}y_{c1} + r_{13}z_{c1} + r_{14}}{r_{31}x_{c1} + r_{32}y_{c1} + r_{33}z_{c1} + r_{34}} \\
&= \left[r_{11}X_1\varepsilon_1 + r_{12}Y_1\varepsilon_1 + r_{13} + \frac{r_{14}}{z_o} (1 - \right. \\
&\quad \left. AX_1\varepsilon_1 - BY_1\varepsilon_1) \right] / \left[r_{31}X_1\varepsilon_1 + r_{32}Y_1\varepsilon_1 \right. \\
&\quad \left. + r_{33} + \frac{r_{34}}{z_o} (1 - AX_1\varepsilon_1 - BY_1\varepsilon_1) \right]
\end{aligned} \quad (27)$$

$$\begin{aligned}
Y_2\varepsilon_2 &= \frac{y_{c2}}{z_{c2}} \\
&= \frac{r_{21}x_{c1} + r_{22}y_{c1} + r_{23}z_{c1} + r_{24}}{r_{31}x_{c1} + r_{32}y_{c1} + r_{33}z_{c1} + r_{34}} \\
&= \left[r_{21}X_1\varepsilon_1 + r_{22}Y_1\varepsilon_1 + r_{23} + \frac{r_{24}}{z_o} (1 - \right. \\
&\quad \left. AX_1\varepsilon_1 - BY_1\varepsilon_1) \right] / \left[r_{31}X_1\varepsilon_1 + r_{32}Y_1\varepsilon_1 \right. \\
&\quad \left. + r_{33} + \frac{r_{34}}{z_o} (1 - AX_1\varepsilon_1 - BY_1\varepsilon_1) \right]
\end{aligned} \quad (28)$$

When the camera parameters $(\alpha_1, \beta_1, \gamma_1)$, $(\alpha_2, \beta_2, \gamma_2)$, ε_1 and ε_2 are available, r_{11} , r_{12} , r_{13} , r_{21} , r_{22} , r_{23} , r_{31} , r_{32} , r_{33} , A , and B are determined. The image registration problem is then equivalent to estimating the three translation parameters $\frac{r_{14}}{z_o}$, $\frac{r_{24}}{z_o}$ and $\frac{r_{34}}{z_o}$ from the correspondences between matched points, which can be achieved by solving a set of linear equations with three unknowns $\frac{r_{14}}{z_o}$, $\frac{r_{24}}{z_o}$ and $\frac{r_{34}}{z_o}$:

$$\begin{aligned}
&\frac{r_{14}}{z_o} (1 - AX_{1i}\varepsilon_1 - BY_{1i}\varepsilon_1) \\
&- \frac{r_{34}}{z_o} (1 - AX_{1i}\varepsilon_1 - BY_{1i}\varepsilon_1) X_{2i}\varepsilon_2 \\
&= (r_{31}X_{1i}\varepsilon_1 + r_{32}Y_{1i}\varepsilon_1 + r_{33}) X_{2i}\varepsilon_2 \\
&- (r_{11}X_{1i}\varepsilon_1 + r_{12}Y_{1i}\varepsilon_1 + r_{13})
\end{aligned} \quad (29)$$

$$\begin{aligned}
&\frac{r_{24}}{z_o} (1 - AX_{1i}\varepsilon_1 - BY_{1i}\varepsilon_1) \\
&- \frac{r_{34}}{z_o} (1 - AX_{1i}\varepsilon_1 - BY_{1i}\varepsilon_1) Y_{2i}\varepsilon_2 \\
&= (r_{31}X_{1i}\varepsilon_1 + r_{32}Y_{1i}\varepsilon_1 + r_{33}) Y_{2i}\varepsilon_2 \\
&- (r_{21}X_{1i}\varepsilon_1 + r_{22}Y_{1i}\varepsilon_1 + r_{23})
\end{aligned} \quad (30)$$

for $i = 1, \dots, N$, where N is the number of matched points.

When the camera information is not available, note that $r_{33}z_o + r_{34}$, the depth of the nadir point in the second camera coordinate system, is always nonzero. We can then rewrite (27-28) as

$$\frac{r_{13} + \frac{r_{14}}{z_o}}{(r_{33} + \frac{r_{34}}{z_o})\varepsilon_2} + \frac{(r_{11} - \frac{r_{14}}{z_o}A)\varepsilon_1}{(r_{33} + \frac{r_{34}}{z_o})\varepsilon_2} X_1$$

$$\begin{aligned}
& + \frac{(r_{12} - \frac{r_{24}}{s_o} B) \epsilon_1}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} Y_1 + \frac{(-r_{31} + \frac{r_{34}}{s_o} A) \epsilon_1 \epsilon_2}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} X_1 X_2 \\
& + \frac{(-r_{32} + \frac{r_{34}}{s_o} B) \epsilon_1 \epsilon_2}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} Y_1 X_2 = X_2 \quad (31)
\end{aligned}$$

$$\begin{aligned}
& \frac{r_{23} + \frac{r_{24}}{s_o}}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} + \frac{(r_{21} - \frac{r_{24}}{s_o} A) \epsilon_1}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} X_1 \\
& + \frac{(r_{22} - \frac{r_{24}}{s_o} B) \epsilon_1}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} Y_1 + \frac{(-r_{31} + \frac{r_{34}}{s_o} A) \epsilon_1 \epsilon_2}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} X_1 Y_2 \\
& + \frac{(-r_{32} + \frac{r_{34}}{s_o} B) \epsilon_1 \epsilon_2}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} Y_1 Y_2 = Y_2 \quad (32)
\end{aligned}$$

Let

$$a_1 = \frac{r_{13} + \frac{r_{14}}{s_o}}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} \quad (33)$$

$$a_2 = \frac{r_{23} + \frac{r_{24}}{s_o}}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} \quad (34)$$

$$a_3 = \frac{(r_{11} - \frac{r_{14}}{s_o} A) \epsilon_1}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} \quad (35)$$

$$a_4 = \frac{(r_{21} - \frac{r_{24}}{s_o} A) \epsilon_1}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} \quad (36)$$

$$a_5 = \frac{(r_{12} - \frac{r_{14}}{s_o} B) \epsilon_1}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} \quad (37)$$

$$a_6 = \frac{(r_{22} - \frac{r_{24}}{s_o} B) \epsilon_1}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} \quad (38)$$

$$a_7 = \frac{(-r_{31} + \frac{r_{34}}{s_o} A) \epsilon_1 \epsilon_2}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} \quad (39)$$

$$a_8 = \frac{(-r_{32} + \frac{r_{34}}{s_o} B) \epsilon_1 \epsilon_2}{(r_{33} + \frac{r_{34}}{s_o} \epsilon_2)} \quad (40)$$

The transform from image1 to image2 can be determined in terms of the eight parameters a_i , $i = 1, \dots, 8$ as

$$X_2 = \frac{a_3 X_1 + a_5 Y_1 + a_1}{-a_7 X_1 - a_8 Y_1 + 1} \quad (41)$$

$$Y_2 = \frac{a_4 X_1 + a_6 Y_1 + a_2}{-a_7 X_1 - a_8 Y_1 + 1} \quad (42)$$

The eight parameters are obtained by solving the linear equations

$$a_1 + X_{1i} a_3 + Y_{1i} a_5 + X_{1i} X_{2i} a_7 + Y_{1i} X_{2i} a_8 = X_{2i} \quad (43)$$

$$a_2 + X_{1i} a_4 + Y_{1i} a_6 + X_{1i} Y_{2i} a_7 + Y_{1i} Y_{2i} a_8 = Y_{2i} \quad (44)$$

for $i = 1, \dots, N$, where N is the number of matched points.

3.3 Algorithm

Overview Figure 3 illustrates the image registration algorithm. Given two images, we first locate two control points to get the north direction in each image. A small number of feature points are then located using a Gabor wavelet model for detecting local curvature discontinuities [20]. The feature points extracted from different

frames are matched using area correlation. Three match verification tests are used to exclude false matches. After the initial matching is achieved, a multiresolution transform-and-correct matching is implemented to obtain a high accuracy registration. At each resolution, the second frame t_2 is first transformed to the coordinates of frame t_1 using the estimated matching parameters and then matching refinement is performed on the feature points of frame t_1 .

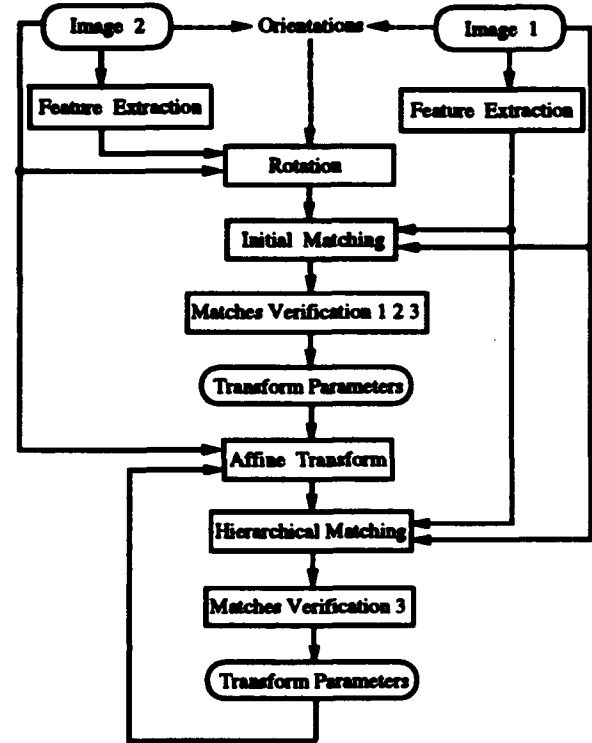


Figure 3: Block diagram of image registration algorithm

There are several variations in the implementation of the above algorithm. First of all, we used two control points to determine the north direction in each image. In real applications, this data is available from the gyrocompass and flight diary. For applications where such data is not available, we use an illuminant direction estimator [21] to get the illuminant directions in each image and then align the images according to known illuminant difference. The use of the illuminant direction estimator is indicated by dashed lines in Figure 3. Secondly, when camera parameters are available, only three translation parameters are required for determination of the image transform; we solve (29–30) to get $\frac{r_{13}}{s_o}$, $\frac{r_{23}}{s_o}$ and $\frac{r_{33}}{s_o}$ and transform the second image to the coordinates of the first camera using (27–28). For applications where camera information is not available, we formulate the transform between the two images in terms of eight parameters and determine the transform parameters by solving (43–44). The second image is then transformed to the coordinates of the first camera using (41–42).

Feature point detection For feature point extraction we use a Gabor wavelet decomposition and local scale interaction based algorithm reported in [20]. The

basic wavelet function used in the decomposition is of the form

$$\Phi(X, Y, \vartheta) = e^{-(X'^2 + Y'^2) + i\pi X'} \quad (45)$$

$$X' = X \cos \vartheta + Y \sin \vartheta \quad (46)$$

$$Y' = -X \sin \vartheta + Y \cos \vartheta \quad (47)$$

where ϑ is the preferred spatial orientation. In our experiments ϑ is discretized into four orientations. The feature points are extracted as the local maxima of the energy measure

$$I(X, Y) = \max_{\vartheta} \{ ||W_2(X, Y, \vartheta) - \gamma W_1(X, Y, \vartheta)|| \} \quad (48)$$

where

$$W_j(X, Y, \vartheta) = f \otimes \Phi(2^{-j_1} X, 2^{-j_2} Y, \vartheta), \quad j = \{j_1, j_2\}.$$

Here j_1 and j_2 are two dilation parameters, and $\gamma = 2^{(j_1 - j_2)}$ is a normalizing factor. In implementing the above algorithm, we further require the energy measure for a feature point to be the maximum in a neighborhood with radius equal to 10 and above a threshold.

Match verification In our algorithm, the initial matching is implemented on 2-D rotation compensated images. Since no further knowledge about the camera parameters is used in the initial matching, false matches due to perspective deformation and similarities between similar objects are inevitable. Automatic exclusion of these false matches is a key to success in image registration. We have used three tests to exclude less reliable matches.

1. **Distance Test:** The translation between the rotation-compensated images should not be larger than certain fraction of the image size. A valid matching pair i should satisfy

$$\begin{cases} d_{ix} = |X_{ir} - X_{il}| \leq \lambda L_x \\ d_{iy} = |Y_{ir} - Y_{il}| \leq \lambda L_y \\ |X_{ir} - X_{il}| + |Y_{ir} - Y_{il}| \leq \kappa \max\{L_x, L_y\} \end{cases} \quad (49)$$

For example, $\lambda = \frac{1}{2}$ and $\kappa = \frac{3}{2}\lambda$.

2. **Variation Test:** The translations used in the correct matches should support each other, i.e.

$$|d_i - \bar{d}| \leq \mu \sigma \quad (50)$$

where d_i is the distance between the matching pair i , \bar{d} and σ are the mean and standard deviation of the distances for all the matched feature pairs, and μ is a threshold, for example $\mu = \sqrt{3}$ for the uniform distribution.

3. **Outlier Exclusion:** The matched feature pairs should satisfy the image transform model. Candidate matching pairs with large residual errors should be excluded. This test also help to exclude matches on building roofs, etc.

3.4 Experimental results

Figure 4 shows the registration of a stereo pair of model board images. Figure 4.a is the image taken by the first camera with parameters $\alpha_w = 40^\circ$ and $\beta = 15^\circ$. The north direction, estimated using two control points, is $\alpha_c = 93.52^\circ$. Figure 4.b is the image taken by the second camera with parameters $\alpha_w = -71^\circ$ and $\beta = 15^\circ$. The north direction, estimated using two control points, is $\alpha_c = 91.40^\circ$. Figure 4.c shows the registration of the second image to the orientation of the first camera when the given camera parameters were used. Figure 4.d shows the registration of the second image to the orientation of the first camera when no camera parameters were assumed.

Figure 5 shows the registration of two off-nadir model board images. Figure 5.a is the image taken by the first camera with parameters $\alpha_w = 14^\circ$ and $\beta = 45^\circ$. The north direction estimated using two control points is $\alpha_c = 274.89^\circ$. Figure 5.b is the image taken by the second camera with parameters $\alpha_w = 50^\circ$ and $\beta = 30^\circ$. The north direction estimated using two control points is $\alpha_c = 10.35^\circ$. Figure 5.c shows the registration of the second image to the orientation of the first camera when the given camera parameters were used. Figure 5.d shows the registration of the second image to the orientation of the first camera when no camera parameters were assumed.

Figure 6 shows the registration of two aerial images. Figure 6.a is the image taken by the first camera, and Figure 6.b is the image taken by the second camera. Since information about the north directions was not available, we used the illuminant direction estimator [21, 22] to get an initial estimate of the camera orientation change. For these two images, no information about the cameras are available. Figure 6.c shows the registration of the second image to the orientation of the first camera.

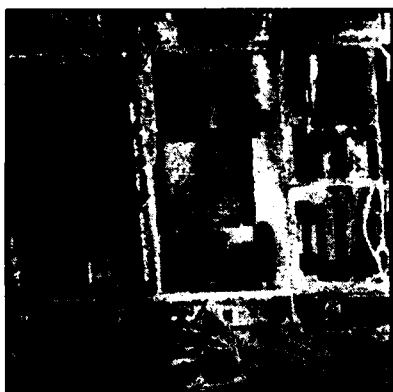
Figure 7 shows the registration of another set of aerial images. Figure 7.a is the image taken by the first camera. Figure 7.b is the image taken by the second camera. Again the illuminant direction estimator was used to get an initial estimate of the camera orientation change. Figure 7.c shows the registration of the second image to the orientation of the first camera.

Acknowledgement

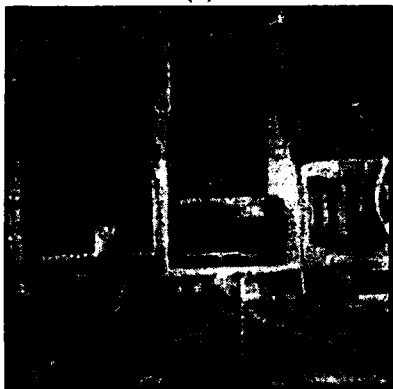
We thank Drs. D. Izraelevitz and T. Moore of TASC for many helpful discussions, and for their contributions to Figure 1 and Section 2.4.

References

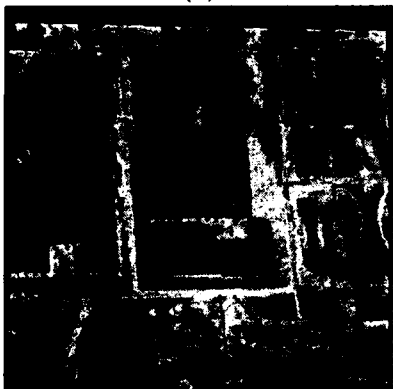
- [1] D. Climenson, "RADIUS Project Office, Model-Supported Exploitation Concepts", unpublished report, Oct. 23, 1992.
- [2] V. Venkateswar and R. Chellappa, "A Hierarchical Approach to Detection of Buildings in Aerial Images", Technical Report CAR-TR-567, Center for Automation Research, University of Maryland, College Park, MD, Aug. 1991.



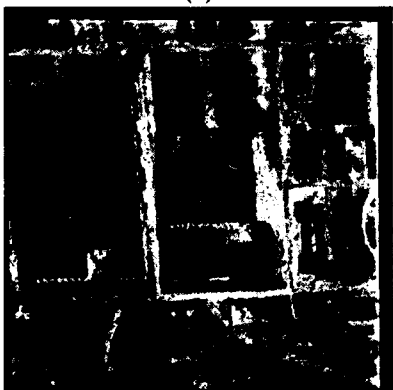
(a)



(b)

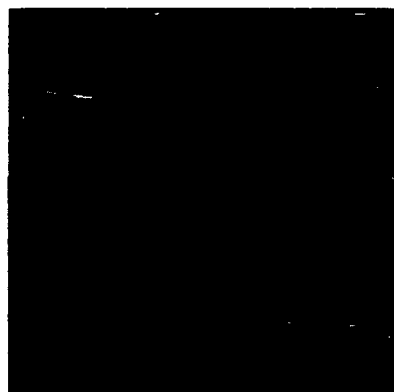


(c)

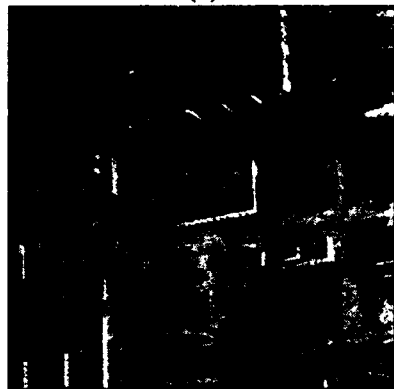


(d)

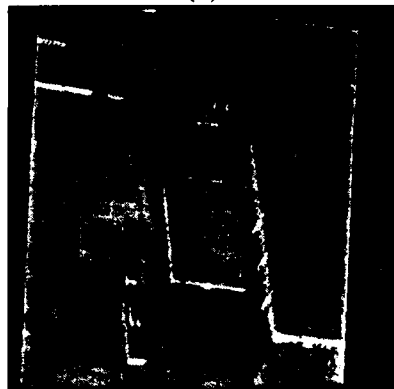
Figure 4: Registration of a stereo pair of model board images. (a) image1. (b) image2. (c) image2 after registration to the orientation of camera1, when camera parameters were used. (d) image2 after registration to the orientation of camera1, when camera parameters were not used.



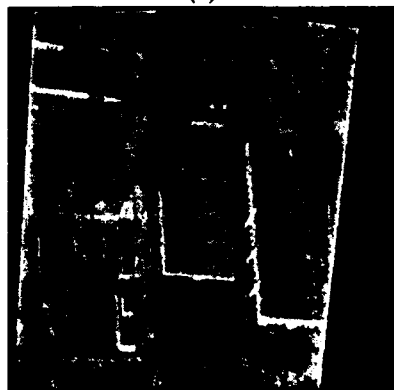
(a)



(b)

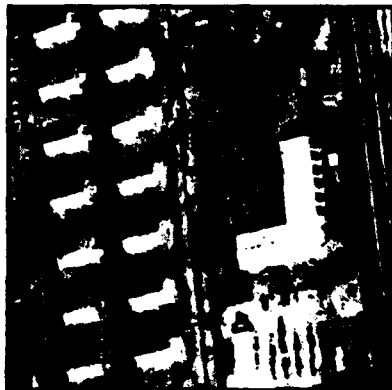


(c)



(d)

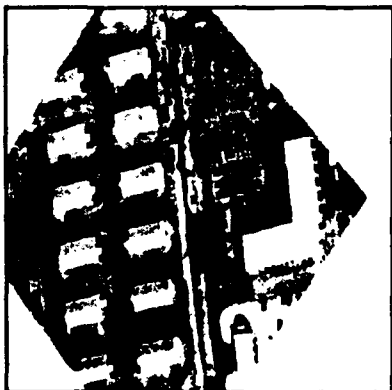
Figure 5: Registration of two off-nadir model board images. (a) image1. (b) image2. (c) image2 after registration to the orientation of camera1, when camera parameters were used. (d) image2 after registration to the orientation of camera1, when camera parameters were not used.



(a)

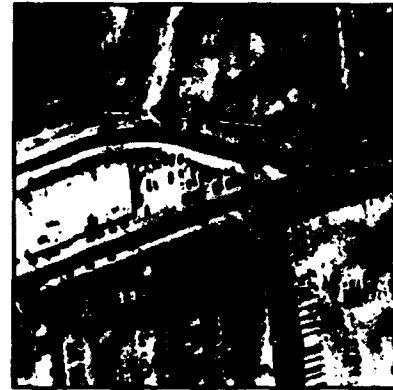


(b)

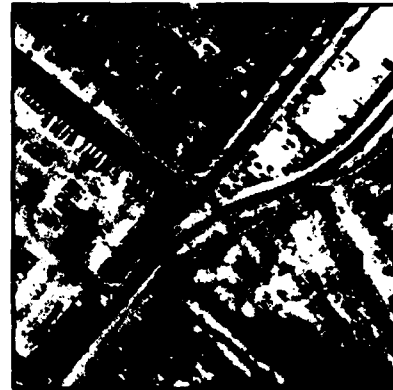


(c)

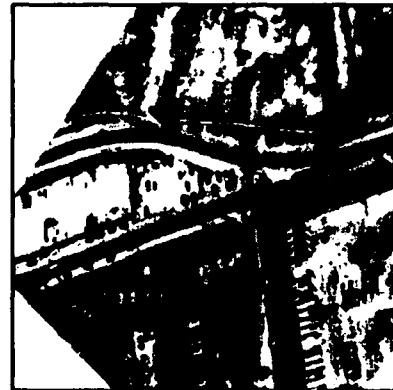
Figure 6: Automatic registration of two aerial images. (a) image1. (b) image2. (c) image2 after registration to the orientation of camera1.



(a)



(b)



(c)

Figure 7: Another example of automatic registration of two aerial images. (a) image1. (b) image2. (c) image2 after registration to the orientation of camera1.

- [3] A. Huertas and R. Nevatia, "Detecting Buildings in Aerial Images", *Computer Vision, Graphics, Image Processing* 41, 131-152, 1988.
- [4] D.M. McKeown, Jr., "Toward Automatic Cartographic Feature Extraction", in *Mapping and Spatial Modeling for Navigation*, L.F. Pau and L. Kanal (eds.), Springer-Verlag, Berlin, 149-180, 1990.
- [5] W.E.L. Grimson and T. Lozano-Perez, "Model-Based Recognition and Localization from Sparse Range or Tactile Data", *Intl. J. Robotics Research* 3, 3-35, 1984.
- [6] R. Mohan and R. Nevatia, "Using Perceptual Organization to Extract 3-D Structures", *IEEE Trans. Pattern Analysis Machine Intelligence* 11, 1121-1139, 1989.
- [7] J. Doyle, "A Truth Maintenance System", *Artificial Intelligence* 12, 231-272, 1979.
- [8] J. de Kleer, "An Assumption-based TMS", *Artificial Intelligence* 28, 127-162, 1986.
- [9] D. Huttenlocher and S. Ullman, "Recognizing Solid Objects by Alignment with an Image", *Intl. J. Computer Vision* 5, 195-212, 1990.
- [10] Y. Lamdan and H. Wolfson, "Geometric Hashing: A General and Efficient Model-based Recognition Scheme", *Proc. Intl. Conf. Computer Vision*, Tarpon Springs, FL, 238-249, 1988.
- [11] T. Silberberg, D. Harwood and L.S. Davis, "Object Recognition Using Oriented Model Points", *Computer Vision, Graphics, Image Processing* 35, 47-71, 1986.
- [12] D. Harwood, R. Prasannappa and L.S. Davis, "Preliminary Design of a Programmed Picture Logic", Technical Report CAR-TR-364, Center for Automation Research, University of Maryland, College Park, MD, June 1988.
- [13] V. Venkateswar and R. Chellappa, "Hierarchical Stereo Matching Using Feature Groupings", Technical Report CAR-TR-556, Center for Automation Research, University of Maryland, College Park, MD, May 1991.
- [14] E. Charniak and D. McDermott, *Introduction to Artificial Intelligence*, Addison-Wesley, Reading, MA, 1985.
- [15] R.M. Stallman and G.J. Sussman, "Forward Reasoning and Dependency-directed Backtracking in a System for Computer-aided Circuit Analysis", *Artificial Intelligence* 9, 135-196, 1977.
- [16] M.L. Ginsberg (ed.), *Readings in Nonmonotonic Reasoning*, Morgan Kaufmann, Los Altos, CA, 1987.
- [17] W. Lukasiewicz, *Non-Monotonic Reasoning*, Ellis Horwood, West Sussex, England, 1990.
- [18] M. Herman and T. Kanade, "Incremental Reconstruction of 3-D Scenes from Multiple, Complex Images", *Artificial Intelligence* 30, 289-341, 1986.
- [19] V. Venkateswar and R. Chellappa, "Extraction of Straight Lines in Aerial Images", *IEEE Trans. Pattern Analysis Machine Intelligence* 16, 1111-1116, Nov. 1992.
- [20] B.S. Manjunath, R. Chellappa, and C. Malsburg, "A Feature-based Approach to Face Recognition," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, Champaign, IL, 373-378, June 1992.
- [21] Q. Zheng and R. Chellappa, "Estimation of Illuminant Direction, Albedo and Shape from Shading," *IEEE Trans. Pattern Analysis Machine Intelligence* 13, 680-702, 1991.
- [22] Q. Zheng and R. Chellappa, "A Computational Vision Approach to Image Registration," in *Proc. Intl. Conf. Pattern Recognition*, The Hague, The Netherlands, 193-197, Aug. 1992.

Employing Contextual Information in Computer Vision

Thomas M. Strat
Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

Abstract

Contextual information is often essential for visual recognition, but the design of image-understanding systems that effectively use context has remained elusive. We describe some of our experiences in attempting to employ contextual information in computer vision systems. By making explicit the built-in assumptions inherent in all computer vision algorithms, an architecture can be designed in which context can influence the recognition process. This paper describes such an architecture for context-based vision (CBV).

1 Introduction

It is generally accepted that the surroundings of an object may have a profound influence on, and in some cases, may be necessary for, visual recognition of the object. What is not so well established is how to design computer vision systems that can exploit such contextual information.

When a human observes a scene, or even studies a photograph, he normally has at his disposal a wealth of information that is not captured by the image alone. For example, if Bob shows Alice some photographs he took, her knowledge that Bob recently vacationed in Hawaii may help her to recognize that the photos were taken there. Any knowledge that Alice has about Hawaii may be useful for recognizing the content of the scene (e.g., that the amorphous landform is actually Diamond Head, and that the vegetation is palmetto bushes and not agave cacti).

An observer can also infer information about the

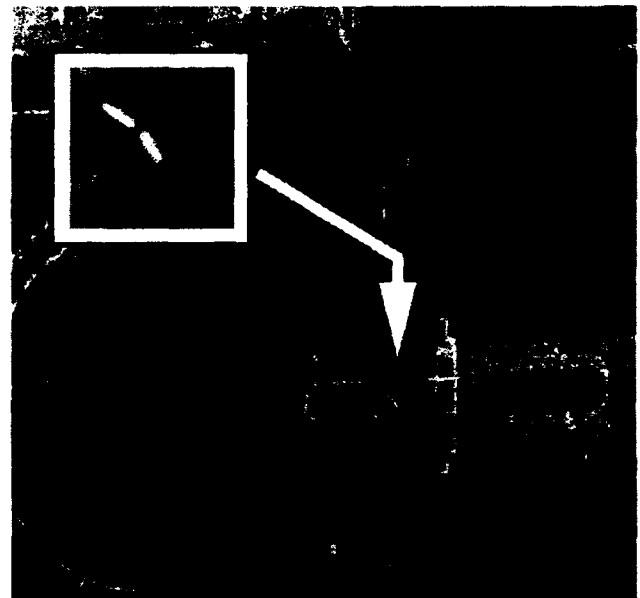


Figure 1: An image in which the use of context is critical to the recognition of some objects.

scene that is then useful for interpreting other parts of the image. For example, given an outdoor scene, usually one can readily determine where the sky is, which direction is vertical, what the weather conditions are, and whether any man-made objects are visible. This information forms part of the context that is available for interpreting the remainder of the scene.

An image such as shown in Figure 1 illustrates the power of contextual information. The inset, a magnified portion of the larger image, displays an object that is difficult to recognize. When the same object is viewed in the context of the intersection of city streets (as in the large image), it is readily recognized as an articulated bus.

¹The work reported here was sponsored by DARPA and monitored by the US Army Topographic Engineering Center under Contract DACA76-92-C-0034.

In this paper, we describe some of our experiences in attempting to employ contextual information in computer vision systems. By making explicit the built-in assumptions inherent in all computer vision algorithms, an architecture can be designed in which context can influence the recognition process. This paper describes such an architecture for context-based vision (CBV).

The first half of the paper summarizes the types of contextual information that are available in image-understanding systems and describes some roles that context can play in the interpretation process. The second half reviews a previously constructed context-based architecture, CONDOR; describes some extensions that are necessary to extend its applicability to semiautomated image understanding (IU); and presents some empirical results of its use in extracting cartographic features.

2 Context-Based Vision

We use the term *contextual information*, or *context* for short, in the broadest sense — to denote any and all information that may influence the way a scene is perceived. Thus, the camera geometry, the image type, the availability of related images, the urgency of observation, and the purpose of image analysis, are all part of the context. A computer vision system, like a human, should be able to use all types of context.

Many authors have used contextual information either implicitly or explicitly in their IU systems, but few have made the representation and use of context a central design feature [4, 5, 7, 13, 21].

The effective use of contextual information can be addressed by considering the design of an overall system architecture, rather than by focusing on individual algorithms. In our view, this can be accomplished by structuring a computer vision system as a composite of many individual algorithms. The contextual information, including the perceptual task and the available imagery, can be used to choose the algorithms most appropriate for each subtask, and can form the basis for evaluating their results. The algorithms can perform independently, but are able to interact through the context that all are controlled by and all contribute to.

The concepts described in this paper are illustrated by examples from two architectures we have

designed:

- CONDOR [17, 18, 19] is a system that analyzes ground-level outdoor imagery of natural environments in the context of a mobile robot application. CONDOR contains an elaborate mechanism for recognizing and labeling natural objects automatically. Because natural objects, unlike man-made objects, are difficult to recognize without consideration of context, analysis of these scenes demands an architecture that makes strong use of contextual information.
- The second architecture is being developed as part of a system for site-model construction using overhead imagery in the RADIUS Project [8]. Unlike CONDOR, this system is designed to be semiautomated — a fact that has implications for both the way in which context can be employed, and for the availability of contextual information. Being a semiautomated design, it relies upon a human operator to replace some of the machinery incorporated in CONDOR and exploits additional contextual constraints supplied by the operator.

3 The Need for Context

The technical problems in using context involve the identification of appropriate representations for the relevant knowledge and the design of an architecture that can effectively invoke this knowledge. A context-based architecture for image understanding must have (among other things) a means for enforcing the assumptions of IU algorithms and a means for accessing relevant information.

3.1 Enforcing Assumptions

Every image-understanding algorithm, by necessity, contains numerous built-in assumptions that limit its range of applicability. For example, some edge-finders work only on binary images, some stereo algorithms cannot handle occlusions, and some road-finders are confounded by strong shadows.

If the results of these algorithms are to be relied upon, the algorithms must not be employed in situations for which their designers did not intend them to be used. It is the context of invocation that

dictates the suitability of an algorithm for a particular task. By explicitly encoding the assumptions and inherent limitations of IU algorithms, one has the potential to control the algorithms by reasoning about the context. Representing assumptions explicitly and matching them to the particular circumstances is one of the keys to using contextual information in a computer vision system.

3.2 Accessing Nonlocal Information

Most IU algorithms also require the use of nonlocal information — data outside the immediate sphere of computation — to assist the interpretation or to control the processing flow. Examples include pixel data that are outside some local processing window, additional images of the same scene, prior facts or expectations that are stored in a map or database, and generic knowledge about the appearance, function, or purpose of objects in a scene. Such information is used by many IU algorithms to compute parameters, to guide search, to cue recognition processes, or to reason about the consistency of an interpretation.

IU algorithms must have access to nonlocal information to aid interpretation. Providing direct access to relevant nonlocal information is another key to using contextual information in a computer vision system.

4 Types of Context

Before describing how contextual information can be represented and used, it is useful to take inventory of the kinds of context that could be considered.

Figure 2 depicts a schematic view of an IU algorithm as a black box. Its explicit inputs are a set of images and some parameters, but it is invoked in the context of an assigned task, a database of facts about the world, and a knowledge base from which additional information about the world can be deduced. Some of its outputs are symbolic descriptions that can also be used to augment the database or knowledge base, or to assign additional tasks for realizing behaviors.

We have found it convenient to divide the range of contextual information into three categories. Additional semantic knowledge may involve contextual information from all three categories.

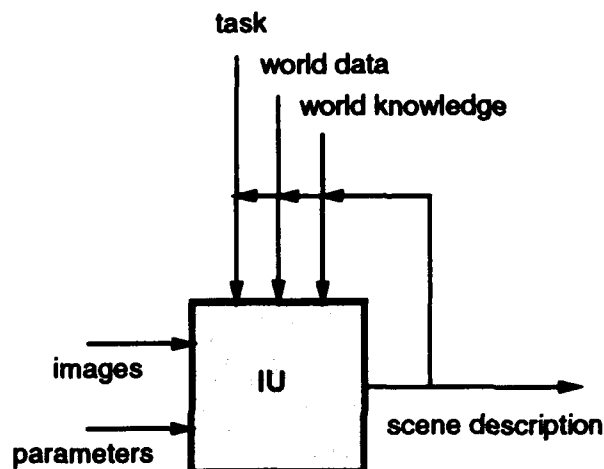


Figure 2: A schematic diagram of an IU algorithm embedded in a vision system.

Physical context — information about the visual world that is independent of any particular set of image acquisition conditions. Physical context encompasses a range of specificity from the very precise “There is a tree at (342, 124)” to the more generic “This area contains a mixed, deciduous forest.” Physical context may also include information about the appearance of scene features in previously interpreted imagery and dynamic information, such as weather conditions and seasonal variations.

Photogrammetric context — information surrounding the acquisition of the image under study. This includes both internal camera parameters (*e.g.*, focal length, principal point, field of view, color of filter) as well as external parameters (*e.g.*, camera location and orientation). We also include the date and time of image acquisition as well as the images themselves.

Computational context — information about the internal state of processing. The computational context can be used to control the processing sequence based on partial recognition results. Different strategies can be used when first initiating the analysis of an image versus filling in the details of a largely completed analysis. The assigned task, the level of automation required, and the available hardware processes are all construed as part of the computational context.

It is worth noting that context may be either established or hypothetical. Tentative conclusions such as "The sky is not visible in this image," and hypothesized facts about the world such as "Assuming that no buildings with peaked roofs are at this site" can be treated as ordinary context to generate hypothetical conclusions.

Just what constitutes contextual information is highly dependent upon the domain of application and the goals of the image-understanding system. CONDOR and RADIUS both involve the delineation and recognition of features of the outdoor world from multiple images. Tables 1-3 detail the types of context used or usable in these applications.

The information in the tables was compiled by examining about one hundred IU algorithms embedded in CONDOR. That list was then augmented by considering additional algorithms that appear to be relevant to the RADIUS site-model construction application. The algorithms considered range from edge-finders [1] to image-segmentation [12], to stereo compilation [2], to snakes [10], to complete object recognition systems [3, 20]. The associated parameters and implicit assumptions for each algorithm were tabulated.

Contextual information may come from a variety of sources, depending on the nature of the application. Some representative sources of contextual information are

- **Database** - Information for use by a vision system may have been previously compiled and stored. Geometric object models, map data, and iconic texture maps are examples.
- **Image header** - Information about the image acquisition is often stored with the image. Camera models, image size and type, and time and date of acquisition are examples.
- **Derived** - Results of earlier IU computation are a valuable source of additional information about a scene.
- **User** - In an interactive or semiautomated scenario, the human operator is also a source of information that can provide context to IU algorithms. This information could range from a general characterization of the image (e.g., urban environment) to a precise, manual extraction of individual features.

Table 1: Physical Context

Geometry	Geometric models of roads, trails, fences, trees, rocks, buildings, railroads, towers, fields, etc. 3D Outline Location Orientation
Photometry/ Radiometry	Albedo Material type Reflectance Surface properties Previous image snippets
Illumination	Sun (azimuth, elevation angles) Haze Cloud cover Shadow contrast
Weather	Temperature Current Precipitation Recent Precipitation Wind speed and direction Season
Geography	Site Terrain type (tundra, desert, ocean, ...) Land use (urban, rural, agricultural, ...) Topography (e.g., Digital Elevation Model) Environmental events (fire, flood, earthquake, war, ...)
Other	Semantic properties (name, use, history, ...)

5 Uses of Context

When an IU algorithm is viewed as a black box as in Figure 3, it is apparent that there are only two opportunities to use contextual information to influence its behavior. At the input end, context can be used to select the best match of image data with IU algorithms and their parameters. At the output end, context can be used to analyze and filter the results.

Choosing algorithms and their parameters: Given an image and a task to be performed, it is necessary to determine the most appropriate algorithm or set of algorithms for accomplishing the task. When the assumptions and limitations of each algorithm have been coded explicitly, it is possible to match their requirements with the context of the present situation, and choose the ones that have (at least) the potential to achieve the desired result. Similarly, a mechanism can be constructed to compute the parameters associated with those algo-

Table 2: Photogrammetric Context

Date and time	
Look angle	Azimuth, elevation, roll
Footprint	Portion of ground observed
Modality	Infrared, color, radar, ...
Multiplicity	Monocular, binocular stereo, multiple, ...
Image size	Pixel dimensions
Image element type	Binary, scalar, vector, complex, ...
Resolution	Ground sample distance (GSD)
Camera model	Focal length, principal point, non-perspective, ...

Table 3: Computational Context

Task	Interpret everything, find tanks, model all buildings, ...
Interactivity	Fully automatic, manual, semiautomatic, batch, continuous interaction, ...)
Urgency	Acceptable processing time
Hardware	Uniprocessor, special-purpose hardware, multiprocessor, ...
Processing state	Just starting, already looked, detailed search, ...

rithms from the available context, although it may be difficult to identify the appropriate computations in advance.

Choosing image data: In some applications, including the CONDOR and RADIUS scenarios, a multitude of imagery is available for analysis. Choosing the subset of images to use can be as critical as the selection of appropriate algorithms. When an algorithm is being considered for invocation, the explicitly coded assumptions can be used to select the images that are best suited to the extraction task being given to that algorithm.

Evaluating results: When IU algorithms have completed their processing, the system has produced a set of results that are best considered as hypotheses. Analysis of the results with the benefit of relevant contextual information can lead to improved interpretations of the imagery. This analysis can take place in several ways — by ranking the hypotheses, by comparing them, by checking their consistency with other hypotheses or with the established context, and so on. In each case, if the analysis software is encoded as a collection of algorithms with explicitly encoded assumptions, one can use the context to choose the algorithms and

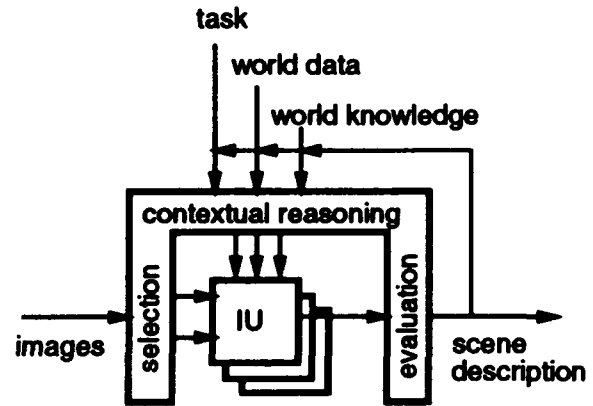


Figure 3: A schematic diagram of a context-based vision system.

control their invocation. Not only does this approach reduce unnecessary computation, but it also simplifies software construction because each algorithm need work only in some narrowly defined context.

6 An Architecture for Context-Based Vision

In the context-based vision paradigm, the invocation of all algorithms is governed by context. Rather than having the control structure and control decisions to be made hard-wired, the process is driven by context.

CONDOR was designed as the perceptual architecture for a hypothetical outdoor robot. Given an image and a possibly extensive database describing the robot's environment, the system is to analyze the image and to augment the world model. CONDOR's recognition vocabulary consists mainly of natural objects such as trees, bushes, trail, and rocks. Because of the difficulty of recognizing such objects individually, CONDOR accepts an interpretation only if it is consistent with its world model. CONDOR recognizes entire contexts, rather than individual objects [17, 18, 19].

6.1 Context Sets

We associate a data structure called a *context set* with each IU algorithm. The context set identifies those conditions that must be true for that algorithm to be applicable. Efficient and effective vi-

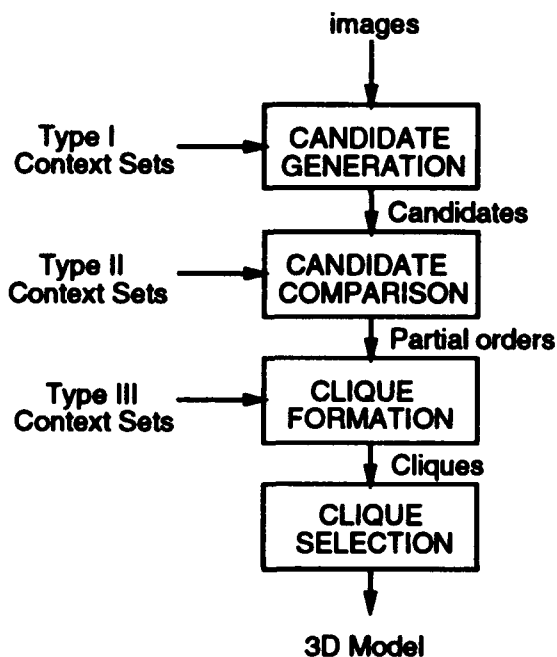


Figure 4: Sequence of Computation in CONDOR.

sual recognition can be achieved only by invoking the IU algorithms in those contexts in which they are likely to succeed.

Formally, a context set is a collection of context elements that are sufficient for inferring some relation or applying some algorithm. A *context element* is a predicate involving any number of terms that refer to the physical, photogrammetric, or computational context of image analysis.

Each algorithm has an associated context set, and is invoked only if its context set is satisfied. A context set is considered to be satisfied only if all its context elements are satisfied.

As an example, consider a simple operator that extracts blue regions to find areas that could be labeled "sky." A context set for this operator might be

{ image-is-color, camera-is-horizontal, sky-is-clear,
time-is-daytime }

The blue-sky algorithm would be unreliable if it were employed in anything but this context.

6.2 Approach

The CONDOR architecture employs three types of algorithms controlled by context sets, as illustrated in Figure 4:

- Type I context sets control IU algorithms that produce candidate (hypothetical) labeled regions.
- Type II context sets control algorithms that compare two candidates and determine if one should be preferred over the other. This step is mainly necessary to limit the combinatorics of finding mutually consistent candidates.
- Type III context sets control algorithms that check if a candidate is consistent with an emerging world model.

For each class in the active recognition vocabulary, all Type I context sets are evaluated. The operators associated with those that are satisfied are executed, producing candidates for each class. Type II context sets that are satisfied are then used to evaluate each candidate for a class, and if all such evaluators prefer one candidate over another, a preference ordering is established between them. These preference relations are assembled to form partial orders over the candidates, one partial order for each class. Next, a search for mutually coherent sets of candidates is conducted by incrementally building cliques of consistent candidates, beginning with empty cliques. A candidate is nominated for inclusion into a clique by choosing one of the candidates at the top of one of the partial orders. Algorithms associated with Type III context sets that have been satisfied are used to test the consistency of a nominee with candidates already in the clique. A consistent nominee is added to the clique; an inconsistent one is removed from further consideration with that clique. Further candidates are added to the clique until none remain. Additional cliques are generated in a similar fashion as computational resources permit. Ultimately, one clique is selected as the best interpretation of the image on the basis of the portion of the image that is explained and the reliability of the operators that contributed to the clique.

The interaction among context sets is significant. The addition of a candidate to a clique may provide context that could trigger a previously unsatisfied context set to generate new candidates or establish new preference orderings. For example, once one bush has been recognized, it is a good idea to look specifically for similar bushes in the image. This tactic is implemented by a candidate-

generation context set that includes a context element that is satisfied only when a bush is added to a clique.

6.3 Representation of Context

We have outlined a paradigm in which the requirements of algorithms are matched against the context of a given situation. To employ this paradigm, it is necessary to have representations for the various categories of contextual information that are to be employed.

The CONDOR system employs the Core Knowledge System (CKS), an object-oriented knowledge/database that was specifically designed to serve as the central information manager for a perceptual system [15]. The CKS provides the ability to store contextual information, and to retrieve it through a vocabulary of spatial and semantic queries. It has the further ability to accommodate conflicting data from multiple sources without corrupting the inference channels. CONDOR uses CKS to store a persistent model of the world, and then uses that model as context for image understanding. Image-understanding results are stored in the CKS and hence are available as context for subsequent processing.

The SRI Cartographic Modeling Environment (CME) provides the primitive representations for modeling the physical objects and their attributes [9]. CME is also used for geometric operations, including coordinate transformation, and for display of imagery and synthetically generated scenes.

6.4 Results

Figure 5(a) depicts an image that typifies those analyzed by CONDOR. After several thousand IU algorithm invocations and construction of 20 cliques, CONDOR's best clique correctly identified six of the trees visible in the image. A perspective view of the grass and trees in the 3D model produced by CONDOR is shown in Figure 5(b).

CONDOR was able to achieve similar results from processing more than 100 images of natural scenes taken within a limited 2-square-mile area. When tasked to analyze images from other natural areas, CONDOR's performance degrades because its contextual knowledge is not totally relevant. This simul-

taneously illustrates the power of using context, as well as the need to encode all contextual constraints that are likely to arise.

7 RADIUS – Site Model Construction

We now turn our attention to the RADIUS project, which is concerned with constructing site models of cultural objects from overhead imagery. Although the specific algorithms to be employed in RADIUS are likely to differ greatly from those in CONDOR, their demands for contextual information are very similar.

The biggest difference between CONDOR and RADIUS is the fact that RADIUS is being designed as a semiautomated system. Accordingly, our design chooses to leave the evaluation of IU results to the human operator. As a result, the Types II and III context sets employed in CONDOR are not necessary. Instead, we concentrate on the construction of Type I context sets for controlling the invocation of IU algorithms. This is particularly appropriate for RADIUS given the wide variety of features to be extracted and the large number of IU laboratories expected to contribute algorithms.

The examples presented here are drawn from an architecture that is being designed to support site model construction for the RADIUS application. The architecture incorporates a large number of generic cartographic feature extraction algorithms; it uses contextual information to identify those most likely to succeed at a given task and to set their associated parameters.

7.1 Model-Based Optimization

While the architecture we have designed is capable of enforcing the contextual constraints of almost any IU algorithm, our initial experiences have focused primarily on employing algorithms from a paradigm known as Model-Based Optimization (MBO).

Specializations of MBO have been referred to by various other terms, including dynamic programming [6], regularization [14], deformable surfaces [22], and snakes [10]. The approach underlying MBO is to express the solution to a feature-extraction problem as a mathematical function of

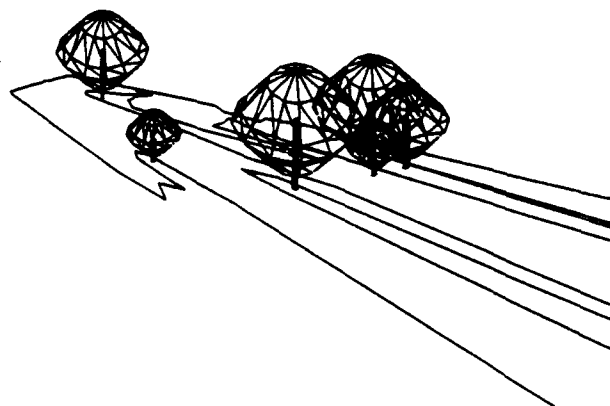


Figure 5: Sample of Processing Results by CONDOR.

some variables, and then to extract the feature from imagery by adjusting the values of the variables to minimize the function. Typically the objective function includes terms that bias the feature's geometry as well as its match with image data. As we have posed it, MBO operators require four parameters: topological primitive, objective function to be minimized, source of initial conditions, and the optimization procedure to be employed. The Context-Based Vision architecture must set these parameters on the basis of known contextual information or (in some cases) human input.

7.2 Context Sets

In CONDOR, Type I context sets are used to specify the conditions that must be met for a given algorithm to be applicable. The context set can also specify the conditions that must be met for a given parameter setting to be useful. For example,

MBO(closed-curve, rectangular-corners,
manual-entry, gradient-descent);

specifies the parameters for an MBO algorithm that could be used to extract roof boundaries under some circumstances. The following context set encodes conditions that are required for the extraction of roofs using that algorithm:

{ image-is-bw, image-resolution \leq 3.0,
interactivity-is-semiautomated }

This context set gives the requirements that must exist for the above MBO algorithm to be applicable and it specifies the suitable parameter values. In the example above for detecting roofs, these parameters have been specified as a closed-curve topology, an objective function preferring rectangular corners, initial boundary provided by manual entry, and the use of a gradient-descent optimization procedure.

In practice, a large number of context sets governing the application of MBO algorithms as well as other algorithms could be constructed and used to implement a cartographic feature-extraction system suitable for site-model construction. It is clear that such a collection could be unwieldy and difficult to maintain. A more structured representation of the context set concept is needed.

7.3 Context Tables

One alternative representation for context sets is the *context table* — a data structure that tabulates the context elements in a more structured fashion. An IU algorithm is associated with each row in the table; each column represents one context element.

The context table is equivalent to a collection of context sets. Conceptually, it provides a more coherent view of the contextual requirements of related algorithms. Applicable algorithms are selected by finding rows for which all conditions are

Table 4: A Context Table

	feature	interactivity	images	resolution	geography	algorithm
1	roof	semiautomated	single BW	≤ 3 meters	—	MBO(topology=closed-curve, obj-fn=rectangular-corners, init>manual-entry, opt=gradient-descent)
2	roof	manual	single	≤ 10 meters	—	CME(primitive=closed-curve)
3	road	semiautomated	single BW	≤ 1 meter	hilly	MBO(topology=ribbon-curve, obj-fn=(smoothness(0.5),continuous, parallel), init>manual-entry, opt=gradient-descent)
4	road	semiautomated	single BW	≤ 10 meters	hilly	MBO(topology=open-curve, obj-fn=(smoothness(0.5),continuous), init>manual-entry, opt=gradient-descent)
5	road	semiautomated	single BW	≤ 1 meter	flat V urban	MBO(topology=ribbon-curve, obj-fn=(smoothness(0.8),continuous, parallel), init>manual-entry, opt=gradient-descent)
6	road	semiautomated	single BW	≤ 10 meters	flat V urban	MBO(topology=open-curve, obj-fn=(smoothness(0.8),continuous), init>manual-entry, opt=gradient-descent)
7	road	manual	single	≤ 10 meters	—	CME(primitive=open-curve)
8	road	manual	single	≤ 1 meter	—	CME(primitive=ribbon-curve)
9	road	semiautomated	single	≤ 2 meters	—	ROAD-TRACKER (control=bidirectional-search, init>manual-entry)

met. Table 4 contains an excerpt of a context table for use in cartographic feature extraction which illustrates the representation.

One drawback to the table representation is its potentially large size. Each algorithm may require many rows to capture the contextual constraints of its various parameter combinations. Its chief value is its organization of contextual information for knowledge-base construction.

7.4 Context Rules

A third alternative for representing context sets is to encode them as rules whose antecedent is the context set, and whose consequent is the applicable algorithm.

For example,

$$\{ \text{image-is-bw, image-resolution} \leq 3.0, \\ \text{interactivity-is-semiautomated} \} \Rightarrow \\ \text{MBO(closed-curve, rectangular-corners,} \\ \text{manual-entry, gradient-descent):}$$

One advantage of encoding the rules as a logic program is that using the logic program interpreter

eliminates the need to devise special machinery to test satisfaction of context sets. The context table of the previous section (Table 4) can be recoded as the roughly equivalent Prolog program given in the Appendix.

A further representational efficiency is possible by collapsing rules with common context elements. For example, the only difference between rules governing Algorithms 3 and 4 and rules governing Algorithms 5 and 6 is the geography term and the value of the smoothness parameter. This dependence could be generalized by additional rules that relate smoothness to geography.

Whatever representation is chosen, it is clear that context sets can be employed in either direction. In the forward direction, the context sets are used to find applicable algorithms. In the opposite direction, the sets can be used for several purposes, including the selection of images on which to invoke a given algorithm. For example, Table 4 shows that the use of an MBO algorithm for finding a roof (Row 1) requires the existence of a monochrome image with 3-meter resolution or better.

7.5 Results

Although the architecture we have described for the RADIUS application is not yet fully functional, we can illustrate its application using the example Table 4.

Figure 6 compares the results of applying an MBO algorithm both within and outside its inherent contextual constraints. Figure 6(a) shows an overhead view of a portion of the Mall in Washington, DC — a flat park area in an urban setting. Figure 6(b) shows an overhead image of a hilly area in the foothills of the Rocky Mountains in Colorado. Both images are shown at approximately the same scale.

The context table in Table 4 can be used to select an algorithm suitable for extracting roads in a semiautomated setting. In the context of the analysis of the Washington DC image, both Algorithm 5 and Algorithm 9 are applicable, but we ignore Algorithm 9 in this example. Algorithm 5 calls for manual entry of the initial curve, which is shown in Figure 6(a). Optimization of this curve using the specified objective function and optimization procedure results in the model depicted in Figure 6(c) — a reasonably accurate extraction of the road.

This algorithm is not applicable to the Rocky Mountain image, because of the different geographical context. If it were applied anyway, optimization of the initial curve shown in Figure 6(b) would result in the curve shown in Figure 6(d) — an extraction that does not follow the road boundaries well.

The context table shows that Algorithm 3 (with its lower smoothness parameter) is applicable for the Rocky Mountain image. Applying it to the same initial curve gives the result depicted in Figure 6(f), a significant improvement over that obtained by Algorithm 5.

Had Algorithm 3 been applied to the Washington DC image (where its context is violated), the result shown in Figure 6(e) would have been obtained — a noticeably poorer delineation of the road than that obtained with a higher smoothness parameter. It is not surprising that the choice of parameters can have a critical effect on the output of an IU algorithm. More important, this example illustrates that contextual information can be successfully used to choose parameter settings.

7.6 Knowledge-Base Construction

The context sets (or context table or context rules) constitute the knowledge base employed by the system. It is clear that the performance of the system will be limited by the accuracy and completeness of the knowledge base. The context sets employed in CONDOR and the context rules being constructed for the RADIUS application are hand-crafted based on *ad hoc* experimentation with available imagery. It is clear that a more automated, or at least a better grounded procedure for constructing the context rules is desirable, both for accommodating a potentially large knowledge base and for extending the domain of competence beyond that originally conceived.

There are several approaches by which the system could learn the most effective context rules. Perhaps the most enticing one for interactive interpretation is one in which the system learns through experience. Whenever a situation arises for which there is no applicable algorithm, or for which all the applicable algorithms give unacceptable results, the human operator has no choice but to edit the result or model the feature by hand, and then continue the site-model construction. Such a manual extraction can serve as the “correct” answer in a supervised learning process. By capturing the context that failed initially, the learning procedure can theoretically compare the results of many algorithms with the “correct” one — whenever there is a sufficiently accurate match, a new context rule can be added. One can also imagine finding a better set of parameters by posing the problem as one for MBO: the algorithm’s parameters can be varied systematically until the best match with “correct” answer is obtained. If the match is sufficiently close, a new context rule with the corresponding parameter settings can be installed.

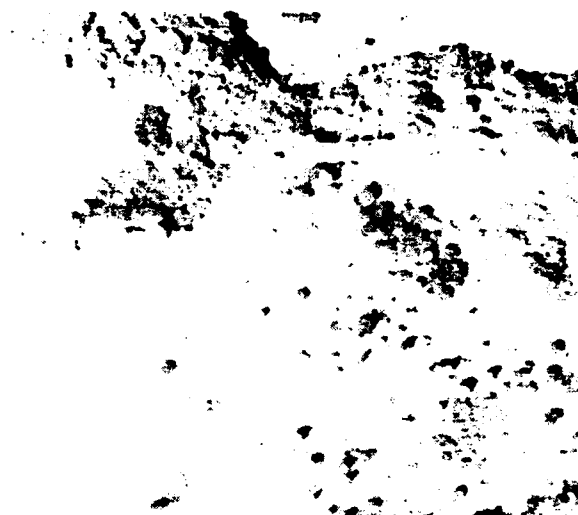
Automating the construction of the context rules is both important and difficult. There are many promising approaches, but none have yet been seriously tried.

8 Summary

We have described some of our experience in applying the CONDOR architecture to the site-model construction task of RADIUS. The semiautomated nature of RADIUS obviates the need for some of the



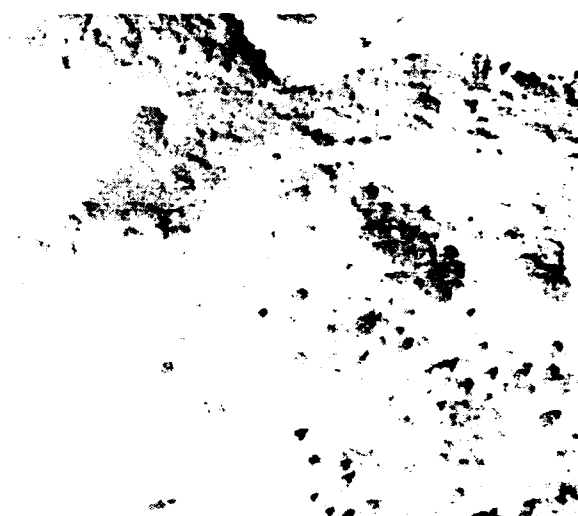
(a)



(b)



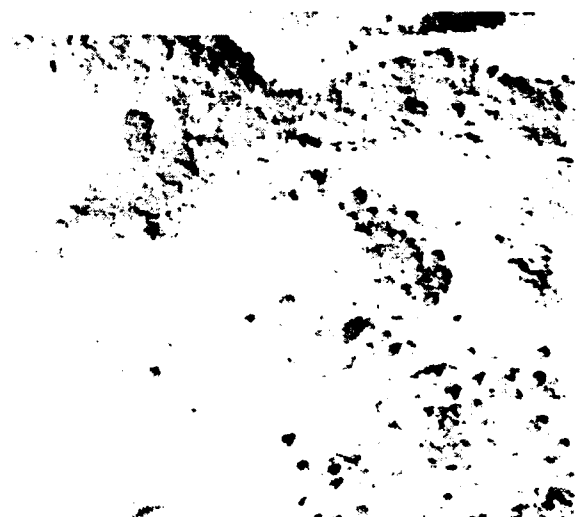
(c)



(d)



(e)



(f)

Figure 6: Context-Based Feature Extraction.

machinery employed in the fully automated design of CONDOR. The availability of a human operator permits access to some kinds of context that were not available to CONDOR, such as the level of interactivity desired, and manual sketches of individual features. The existence of a human to review and edit the IU results offers the opportunity to use a supervised learning scheme to improve the quality of the knowledge base or to extend its range of competence.

The large number of features and wide range of imaging conditions that must be considered for site-model construction in RADIUS stress the context set representation employed in CONDOR. While context sets were adequate for the knowledge base of CONDOR, it has been necessary to consider more effective representations that will extend to the requirements of site-model construction. Two new constructs — context tables and context rules — offer a more systematized organization for the context knowledge base that should facilitate its construction. These representations offer additional economies in both storage and computation that may be vital to implementation of large systems. The symmetry of context tables and rules encourages their use in either direction: to select algorithms and set their parameters, or to describe the conditions that must be satisfied for a given algorithm to be applicable. This final capability raises the possibility of using context rules to choose the most appropriate images for interpretation.

Acknowledgments

I am indebted to Marty Fischler for the numerous discussions that motivated and shaped much of this work. Thanks also to Pascal Fua for the use of his snake algorithms and to Lynn Quam for supplying the Cartographic Modeling Environment which facilitated the implementation and experimentation enormously.

Appendix

The following Prolog program² is roughly equivalent to the context table depicted in Table 4.

²More compact programs are possible.

```
% alg(Name, Parameters) :-
%
% alg specifies the applicable functions and their
% appropriate parameter settings for use in a
% prescribed context
% Name is a symbol denoting the function to be invoked
% Parameters is a sequence of parameters whose format
% depends on the function

alg(mbo, [closed-curve, obj-fn(rectangular-edges),
manual-entry, gradient-descent]) :-
    object-type(roof),
    site(Site),
    interactivity(semiautomated),
    image-site(Image, Site),
    modality(Image, bw),
    image-resolution(Image, GSD),
    GSD =< 3.0 .

alg(cme, [closed-curve]) :-
    object-type(roof),
    site(Site),
    interactivity(manual),
    image-site(Image, Site),
    image-resolution(Image, GSD),
    GSD =< 10.0 .

alg(mbo, [ribbon-curve,
obj-fn(smoothness(0.5), continuous, parallel),
manual-entry, gradient-descent]) :-
    object-type(road),
    site(Site),
    interactivity(semiautomated),
    image-site(Image, Site),
    modality(Image, bw),
    image-resolution(Image, GSD),
    GSD =< 1.0,
    site-geography(Site, hilly) .

alg(mbo, [open-curve,
obj-fn(smoothness(0.5), continuous),
manual-entry, gradient-descent]) :-
    object-type(road),
    site(Site),
    interactivity(semiautomated),
    image-site(Image, Site),
    modality(Image, bw),
    image-resolution(Image, GSD),
    GSD =< 10.0,
    site-geography(Site, hilly) .

alg(mbo, [ribbon-curve,
obj-fn(smoothness(0.8), continuous, parallel),
manual-entry, gradient-descent]) :-
    object-type(road),
    site(Site),
    interactivity(semiautomated),
    image-site(Image, Site),
    modality(Image, bw),
    image-resolution(Image, GSD),
    GSD =< 1.0,
    site-geography(Site, flat) .

alg(mbo, [open-curve,
obj-fn(smoothness(0.8), continuous),
manual-entry, gradient-descent]) :-
    object-type(road),
    site(Site),
    interactivity(semiautomated),
    image-site(Image, Site),
```

```

modality(Image, bw),
image-resolution(Image, GSD),
GSD =< 10.0,
site-geography(Site, flat) .

alg(cme, [open-curve] ) :-
  object-type(road),
  site(Site),
  interactivity(manual),
  image-site(Image, Site),
  image-resolution(Image, GSD),
  GSD =< 10.0 .

alg(cme, [ribbon-curve] ) :-
  object-type(road),
  site(Site),
  interactivity(manual),
  image-site(Image, Site),
  image-resolution(Image, GSD),
  GSD =< 1.0 .

alg(road-tracker,
  [bidirectional-search, manual-entry] ) :-
  object-type(road),
  site(Site),
  interactivity(semiautomated),
  image-site(Image, Site),
  image-resolution(Image, GSD),
  GSD =< 2.0 .

```

References

- [1] Ballard, Dana H., and Christopher M. Brown, *Computer Vision*, Chapter 3, Prentice-Hall, Inc., 1982.
- [2] Barnard, Stephen T., and Martin A. Fischler, "Computational Stereo," *Computing Surveys* 14(4), pp. 553-572 (December 1982).
- [3] Binford, Thomas O., "Survey of Model-Based Image Analysis Systems," *Int. J. Robotics Research*, Vol. 1, pp. 18-64 (1982).
- [4] Clément, Véronique, Gérard Giraudon, Stéphane Houzelle, and Fadi Sandakly, "Interpretation of Remotely Sensed Images in a Context of Multisensor Fusion using a Multi-Specialist Architecture," Institut National de Recherche en Informatique et en Automatique (INRIA), Research Report, Sophia Antipolis, France (October 1992).
- [5] Draper, Bruce A., Robert T. Collins, John Brolio, Allen R. Hanson, and Edward M. Riseman, "The Schema System," *Internat. Jour. Computer Vision*, 2(3), pp. 209-250 (January 1989).
- [6] Fischler, M.A., and R.A. Elschlager, "The Representation and Matching of Pictorial Structures," *IEEE Trans. on Comp.* 22(1), pp. 67-92 (January 1973).
- [7] Garvey, Thomas D., "Perceptual Strategies for Purposive Vision," Ph.D. Dissertation, Department of Electrical Engineering, Stanford University, Stanford, CA (December 1975).
- [8] Gerson, Donald J., "RADIUS: The Government Viewpoint," *Proceedings, DARPA Image Understanding Workshop*, San Diego, CA, pp. 173-176 (January 1992).
- [9] Hanson, Andrew J., and Lynn Quam, "Overview of the SRI Cartographic Modeling Environment," *Proceedings: DARPA Image Understanding Workshop*, Cambridge, MA, pp. 576-582 (April 1988).
- [10] M. Kass, A. Witkin and D. Terzopoulos, "Snakes: Active Contour Models," *Internat. Jour. Computer Vision*, 1(4), pp. 321-331 (1987).
- [11] Lawton, D.T., T.S. Levitt, C. McConnell, and J. Glicksman, "Terrain Models for an Autonomous Land Vehicle," in *Proc. 1986 IEEE International Conference on Robotics and Automation*, San Francisco, CA, pp. 2043-2051 (April 1986).
- [12] Y. G. Leclerc, "Constructing Simple Stable Descriptions for Image Partitioning," *Internat. Jour. Computer Vision* 3(1), pp. 73-102 (May 1989).
- [13] McKeown, David M., Jr., W.A. Harvey, Jr., and J. McDermott, "Rule-Based Interpretation of Aerial Imagery," *IEEE Trans. Pattern Analysis and Machine Intelligence* 7(5), pp. 570-585 (September 1985).
- [14] Poggio, Tomaso, Vincent Torre, and Christof Koch, "Computational Vision and Regularization Theory," *Nature* 317(26), pp. 314-319 (September 26, 1985).
- [15] Strat, Thomas M., and Grahame B. Smith, "The Core Knowledge System," Technical Note 426, Artificial Intelligence Center, SRI International, Menlo Park, CA (October 1987).
- [16] Strat, Thomas M., and Grahame B. Smith, "A Knowledge-Based Information Manager for Autonomous Vehicles," Chapter 1 in Su-shing Chen (Ed.), *Image Understanding in Unstructured Environments*, World Scientific Publishing Co., Singapore, pp. 1-39 (1988).
- [17] Strat, Thomas M. and Martin A. Fischler, "Natural Object Recognition: A Theoretical Framework and Its Implementation," in *Proceedings, IJCAI-91*, Sydney, Australia, pp. 1264-1270 (August 1991).
- [18] Strat, Thomas M., and Martin A. Fischler, "Context-Based Vision: Recognizing Objects Using Both 2D and 3D Imagery," *IEEE Trans. Pattern Analysis and Machine Intelligence* 13(10), pp. 1050-1065 (October 1991).
- [19] Strat, Thomas M., *Natural Object Recognition*, Springer-Verlag, New York, 1992.
- [20] P. Suetens, P. Fua, and A.J. Hanson, "Computational Strategies for Object Recognition," *ACM Computing Surveys* 24(1) (March 1992).
- [21] Tenenbaum, Jay M., Harry G. Barrow, Robert C. Bolles, Martin A. Fischler, and Helen C. Wolf, "Map-Guided Interpretation of Remotely-Sensed Imagery," *Pattern Recognition and Image Processing*, pp. 610-617 (February 1979).
- [22] Terzopoulos, D., A. Witkin, and M. Kass, "Constraints on Deformable Models: Recovering 3D Shape and Non-rigid Motion," *AI Journal* 36, pp. 911-123 (1988).

Research in Automated Analysis of Remotely Sensed Imagery

David M. McKeown, Jr.
Wilson A. Harvey
Stephen J. Ford

Steven Douglas Cochran
J. Chris McGlone

Stephen J. Gifford
Michael F. Polis
Jefferey A. Shufelt

Digital Mapping Laboratory
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891

Abstract

This paper presents an overview of our program of research toward the automated analysis of remotely sensed imagery. Research results in the areas of photogrammetry, stereo analysis, automatic building extraction, digital terrain modeling, knowledge-based systems, and multispectral image analysis are presented.¹

1. Introduction

The automated compilation of man-made and natural terrain in urban or *built-up* areas has been the focus of our research for a number of years. Built-up areas provide some of the most difficult and time consuming tasks for the cartographic community, and provide a rich environment of varied structures and natural terrain features to test the robustness of new approaches to computer vision. The theme of our research is to understand the computational aspects of automated recovery of three-dimensional scene information using a variety of image domain cues. These cues include the analysis of cast shadows, stereo matching, geometric models, and structural descriptions based upon analysis and combination of low-level image-based features. We look for opportunities to augment traditional computational vision techniques with domain knowledge since it is

clearly used by both cartographers and imagery analysts in a variety of tasks ranging from mapping to environmental land use analysis to natural resource inventory.

In this paper we provide a status report in a variety of research activities. Section 2 describes recent work in the application of rigorous photogrammetric methods to image orientation and building extraction within the context of the DARPA RADIUS program. We also describe recent results using our stereo matching systems [McKeown and Hsieh 92, Hsieh et al. 92] on model board imagery. A companion paper in this proceedings describes the incorporation of vanishing point geometry into the BABE building extraction system [McGlone and Shufelt 93]. Such geometric analysis is a key requirement for cartographic feature analysis for oblique imagery, particularly in the fusion of results from multiple views.

Section 3 describes new research in developing symbolic and geometric descriptions for buildings using multiple cues such as stereo, shadow analysis, and monocular building detection. The goal is to detect those regions in the disparity map created by stereo matching that correspond to buildings. Given that structures may appear on rolling terrain, and that stereo analysis rarely constructs an error-free model of the scene, simple techniques based upon region analysis must be augmented with other sources of information.

Section 4 details an experiment in measuring the effectiveness of human-computer interaction to aid in building detection. While most of our research has focused upon automated end-to-end analysis, there is a role for user interaction at various stages of the cartographic process. Some preliminary results are presented in interactive building selection using a simple pointing method.

¹ This work was sponsored by the Defense Advanced Research Projects Agency under Contract DACA76-92-C-0036, by the U.S. Army Topographic Engineering Center and the Defense Advanced Research Projects Agency under Contract DACA72-91-C-0014, and by the Air Force Office of Scientific Research under Contract F49620-92-J-0318. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the U.S. Army Topographic Engineering Center, the Air Force Office of Scientific Research, or the United States Government.

Section 5 describes a continuation of previously reported research toward the development of improved techniques for a terrain representation called a triangular irregular network (TIN) [Polis and McKeown 92]. A new point selection algorithm is described and results for the generation of a 2500 kilometer square area of the National Training Center (NTC), Fort Irvin, California, are shown.

Research on knowledge acquisition and refinement for rule-based systems used for the interpretation of aerial imagery is reported on in Section 6. Using a large production system architecture, SPAM, we report on ongoing research in the evaluation of the utility of various sources of knowledge for airport scene analysis.

Finally, in Section 7 we briefly describe current research in multispectral analysis to determine surface material properties as a knowledge source of built-up area segmentation and cartographic feature extraction. A detailed description of performance evaluation for two classification techniques can be found in a companion paper [Ford et al. 93] in this volume.

2. Photogrammetric approach

Our goal is to apply rigorous photogrammetric methods in several areas of research, particularly to improve the extraction of geometric cues, and to relate partial object descriptions across multiple images. One of our first applications has been the incorporation of vanishing point geometry into the BABE building extraction system, as discussed in [McGlone and Shufelt 93] in this volume.

In this section we discuss our research using the RADIUS modelboard imagery. Under the RADIUS program a set of images of a synthetic industrial site, represented by a scale model, were created and distributed to the RADIUS community. These images differ from more typical mapping photography used in cartographic research in that they are taken from oblique angles rather than near-nadir (down looking) mapping cameras. Along with the imagery a set of ground control points with known modelboard coordinates were distributed.

Using this imagery we have addressed two major areas. The first is the implementation of a rigorous central projection camera model and the solution for the camera parameters for the modelboard imagery. The second is the evaluation of our current stereo matching techniques developed for traditional mapping imagery using the modelboard imagery.

Image ID	# Ground Points	# Image Points	RMS Image Residuals
J3	76	76	2.2 pixels
J4	67	67	1.8
J5	65	65	2.4
J6	77	77	2.3
J7	81	81	2.4
J3 — J7	111	366	2.3

Table 1: RADIUS modelboard resection results.

2.1. Modelboard image resection

Our work to date has been focused on an initial set of eight images, J1 through J8, of the modelboard industrial site. Figures 1 and 2 are two overlapping areas, from images J5 and J4 respectively, and illustrate typical scene content. In order to obtain valid position and orientation parameters for the images we implemented a standard photogrammetric resection procedure.

A relatively large number of modelboard control points (70-80) were measured in each of five images, J3 through J7. We scaled the RADIUS modelboard control point coordinates into world units using the modelboard scale information, given as 1:500. In order to better integrate with our existing landmark database software [McKeown 87] we transformed the modelboard coordinates into pseudo geodetic (latitude-longitude) coordinates. The coordinate origin of the modelboard imagery was taken to be somewhere in central Kansas. For each of the images we performed an individual image resection to establish an error measure based upon RMS image displacement of the measured points. In addition we performed a simultaneous block adjustment of the modelboard images using all of the measured points. Simultaneous resection of the images in the same adjustment allows better error detection, due to the higher redundancy in the solution, and gives orientation parameters that are more consistent between images.

Results of the individual and simultaneous adjustments of images J3 through J7 are shown in Table 1. One can see a fairly consistent residual error of about 2.3 pixels in these resections. Further refinements of our camera model may improve this situation, but at the current scale of the modelboard photography these errors correspond to about a three foot displacement in ground position. Sources of error include uncertainty in the modelboard ground control locations, errors in the measurement of these points

in each of the modelboard images, and unmodeled distortions resulting from the image formation process. It remains to be seen how such errors will effect the accuracy of cartographic feature descriptions, such as buildings, whose models are composed from partial object descriptions acquired from multiple views of the scene.

An important output of the resection solution is the precision information obtained on the orientation parameters, which can be propagated to estimate the precision on calculated ground coordinates, distances, or heights. These precision estimates will in turn allow us to more meaningfully control and merge various operations.

The image resection parameters are used pervasively in our research. For building extraction from the oblique aerial imagery, the vanishing point information is directly calculated and exploited as described in [McGlone and Shufelt 93]. In the stereo processing the image orientation parameters are used to precisely resample the images into epipolar geometry and to calculate elevations and heights in the scene. The incorporation of precise camera models, resection information, and precision information into other applications is now in progress.

2.2. Stereo analysis of the RADIUS modelboard imagery

The RADIUS modelboard imagery presents several new complexities in the interpretation of aerial imagery. The emphasis on oblique views breaks some of the basic assumptions built into processes that analyze and interpret near-vertical stereo pairs. In order to establish a performance baseline for our stereo analysis systems we processed the two stereo pairs (J4-J5) and (J6-J7) found in the initial release of the RADIUS modelboard dataset. We used our standard orientation methods developed for near-vertical imagery taken along a single flightpath [Perlant and McKeown 90] as well as our new image orientation system based upon the resection results reported in the previous section. Both pairs, (J4-J5) and (J6-J7), are relatively wide angle stereo, with convergence angles of 60 and 25 degrees, respectively. For the examples in this section we show results using the 60 degree pair because it represents an extreme case with respect to our previous work.

2.2.1. Stereo matching and refinement

Most stereo systems in cartographic analysis assume that the stereo pair is in a collinear epipolar geometry. We use two independent stereo matching systems of this type. The first, S1, is an area-based method that provides good figural-continuity and captures a sense of foreground and background. It works in a hierarchical coarse-to-

fine fashion in order to capture as much global continuity as possible while retaining a locally-based process. Its results are best in smooth textured areas, but tends to smooth (blur) abrupt changes in depth.

The second, S2, is a feature-based method that provides a more accurate estimate at a few points—especially near depth discontinuities, but requires interpolation to “fill in the gaps.” This process also uses a hierarchical coarse-to-fine approach, but matches “waveform” features across (epipolar) scanlines rather than a correlation window. To remove false matches this process uses a inter-/intra-scanline consistency check [McKeown and Hsieh 92].

The results of the two stereo processes are refined using a monocular segmentation of the original intensity image into homogeneous regions. This process first merges the disparity results from each stereo method using a common estimate of “goodness” to select the best match; however, if there is a large disagreement between the two methods, then both estimates are suppressed. Within each region of the segmentation, which is assumed to represent a single continuous patch of surface, the disparity values are averaged and the outliers are removed. Two different segmentations are used to limit the formation of artifacts during this process [McKeown and Perlant 92].

2.2.2. Generation of epipolar geometry

Epipolar resampling, that is, resampling a stereo pair of images so that the epipolar lines run along the rows of the image, is a requirement for our stereo matchers, as it is for most existing computer vision systems applied to aerial imagery. Unfortunately, the resampling that is typically performed uses approximate warping techniques that may be adequate for vertical images but may fail for imagery with severe obliquity. We have implemented a rigorous epipolar reprojection routine that transforms a given stereo pair into the required geometry using the full orientation parameters for the images.

As an experiment we generated epipolar aligned imagery using two different techniques. First, we established a baseline registration by performing a relative orientation of the RADIUS modelboard imagery by finding common scene points in each of the stereo pairs. A polynomial orientation was performed giving an approximately collinear epipolar alignment [Perlant and McKeown 90]. The second orientation was performed using a rigorous epipolar reprojection based upon the modelboard resection.



Figure 1: Image J5 (left view).

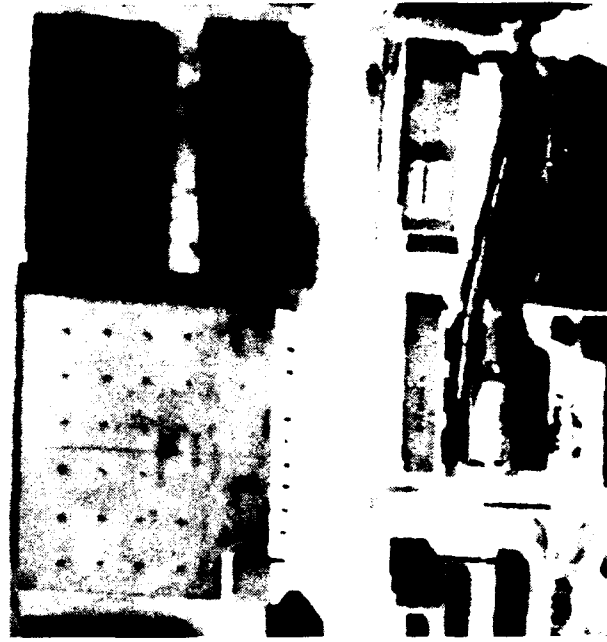


Figure 2: Image J4 (right view).



Figure 3: Refined S1 disparity results using polynomial orientation.

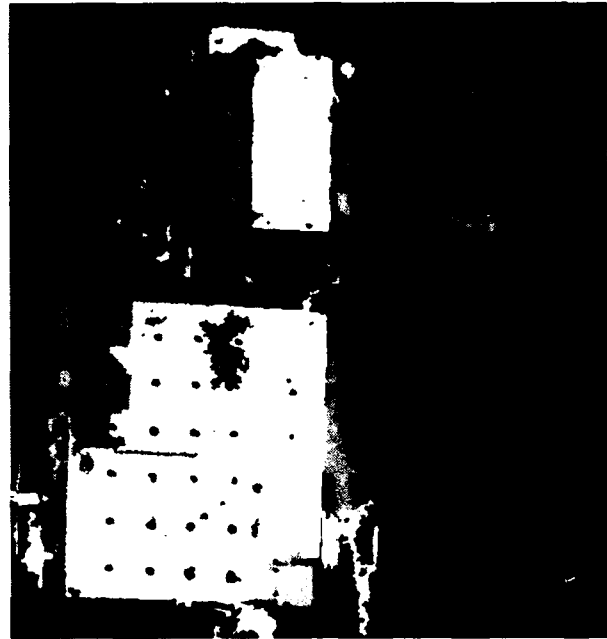


Figure 4: Refined S2 disparity results using polynomial orientation.

2.2.3. Modelboard stereo results

Both S1 and S2 were run using both the polynomial orientation and the resection reprojection on the RADIUS J4 and J5 stereo pair. Figures 1 and 2 show the left and right image pairs. Figures 3 and 4 show the stereo disparity results after refinement using the polynomial orientation. The disparity results are encoded such that bright areas are higher than dark areas.

Using the polynomial orientation one can easily see areas of mismatch, especially for the area-

based S1 process. This was mostly due to the lack of a precise alignment of the epipolar lines. In addition, the large number of occluded regions caused several mismatches by the feature-based S2 matcher. In many cases the correct match between features had an opposite intensity contrast, which violates one of the current S2 constraints. This was less an issue of registration and due more to the imaging geometry.

Figure 5 shows the reprojection of the original left image in Figure 1 such that the camera axis is

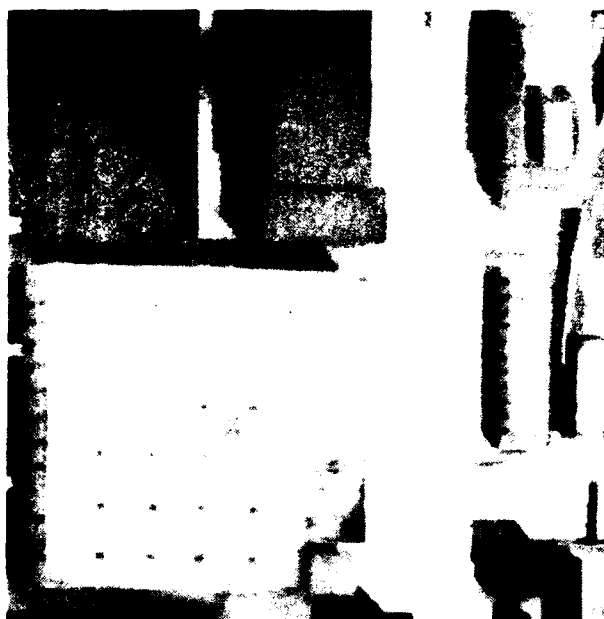


Figure 5: Resampled Image J5 (left view) using full resection.

perpendicular to the stereo baseline. Both the left and right images were reprojected into epipolar alignment. One can notice a change in shape of the scene due to the obliquity and the angle between the original camera axis and the stereo baseline. In this case the change was minimal since the baseline was nearly horizontal.

Figure 6 shows the result of S2 matching and refinement using the resection orientation. Although the results in Figures 4 and 6 are not directly comparable due to the reprojection, one can see significantly more structure in the buildings in the upper left corner of the scene. Figures 7 and 8 show perspective views of the modelboard reconstruction and also highlight some of the differences between the two orientation techniques. Quantitative analysis of the stereo accuracy along the lines of [Hsieh et al. 92] will be performed over a set of test cases.

2.3. Open issues

As discussed in Section 2.1 we have applied a rigorous photogrammetric approach to the problem of obtaining a more exact collinear epipolar alignment of the stereo images. We still need to determine how much tolerance our stereo algorithms exhibit with oblique imagery. With the larger angle oblique images, we will need to consider methods to deal with the large occluded areas and the large baseline-to-range ratio.

We have observed that the stereo refinement process greatly improved the disparity results in both of the modelboard image tests. We plan to introduce additional sources of information in the stereo process, such as wall and roof hypotheses



Figure 6: Refined S2 disparity results using full resection.

generated by monocular analysis, in order to help guide and refine the stereo matching.

3. Building extraction from stereo

The interpretation of stereo disparity maps to detect and delineate manmade structures contained within is a difficult problem. Our recent research has been addressing the detection and extraction of buildings using stereo analysis together with monocular cues. The goal is to produce full three-dimensional models of complex buildings for site model construction and update. Our approach is to apply the cooperative-methods paradigm starting with the results generated by the stereo analysis of a pair of aerial images and, together with monocular cues, mark those areas of the image that appear to be structures.

Our first step is to obtain a set of refined stereo estimates of the scene. This is obtained by using the S1 and S2 stereo matching systems coupled with disparity map refinement as described by [McKeown and Perlant 92]. In the course of the stereo refinement process an intensity segmentation is produced. This segmentation is used as the basis for subsequent processing.

The second step is to merge those segmented regions that have approximately the same disparity and that are adjacent. Next, those (merged) regions having a significantly greater disparity than their neighbors are selected. The rule applied in this step is liberal in the sense that we would rather produce a few false positives that miss buildings at this point.

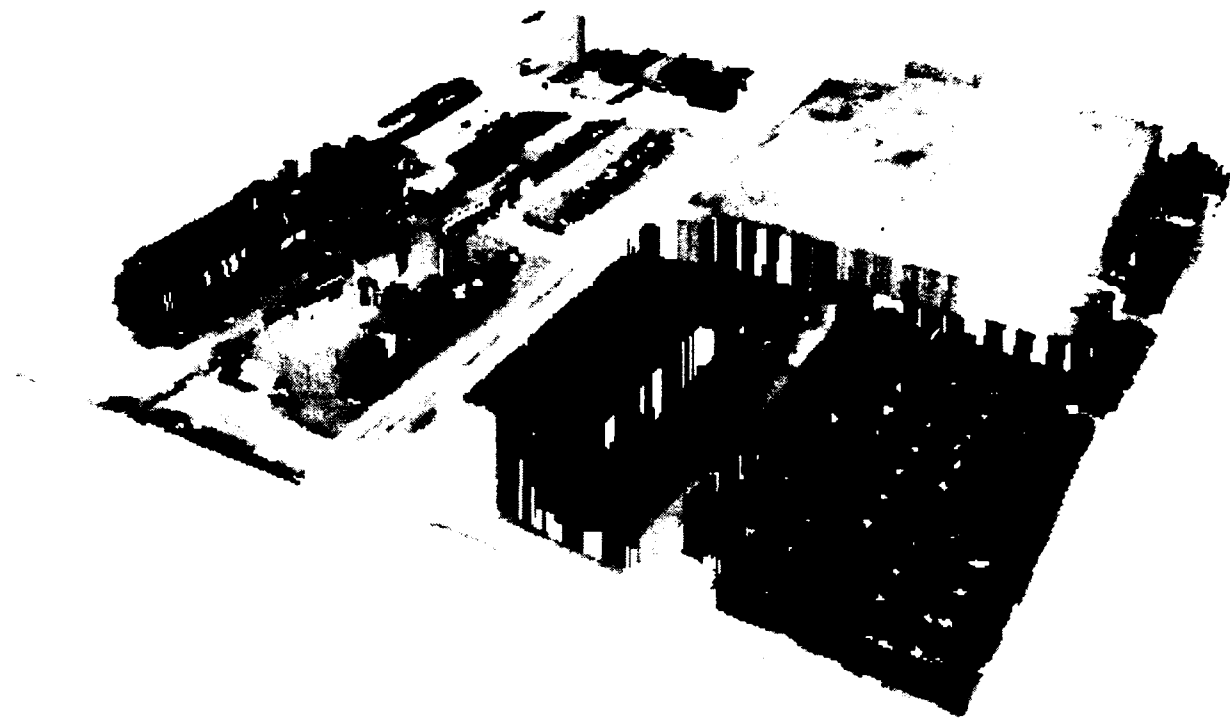


Figure 7: S2 stereo results (J5-J4) using polynomial orientation.

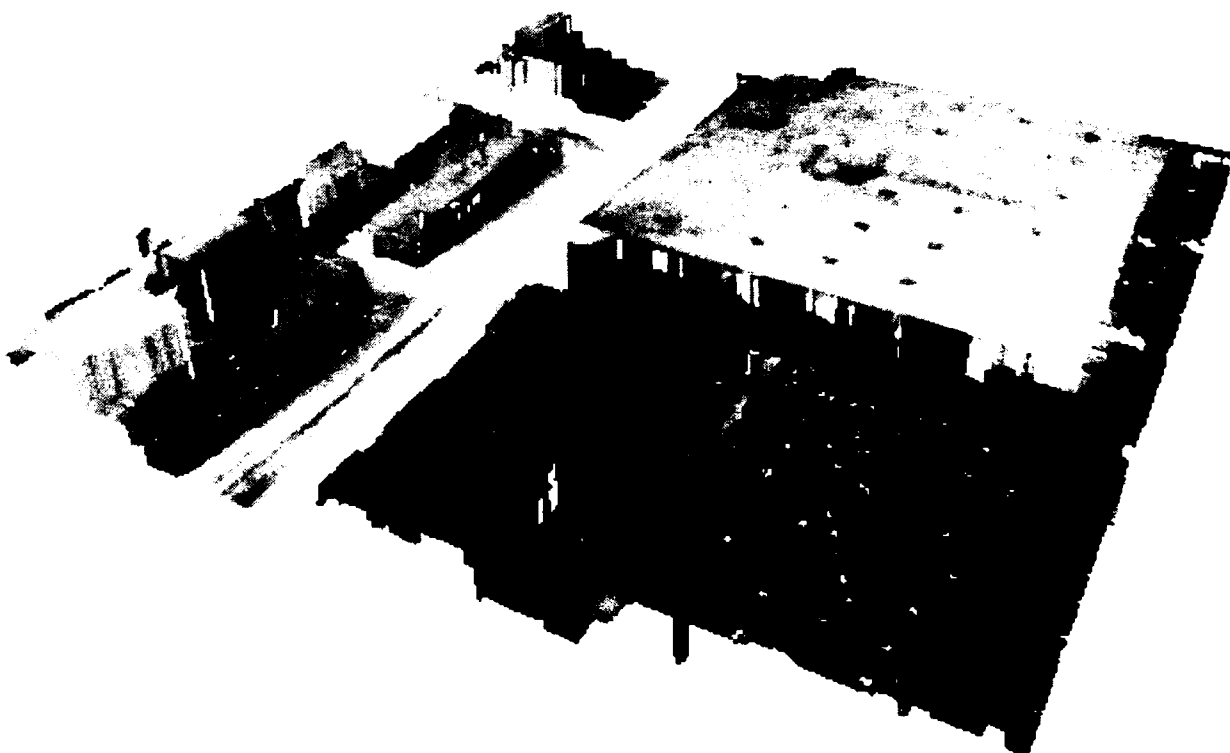


Figure 8: S2 stereo results (J5-J4) using full resection.

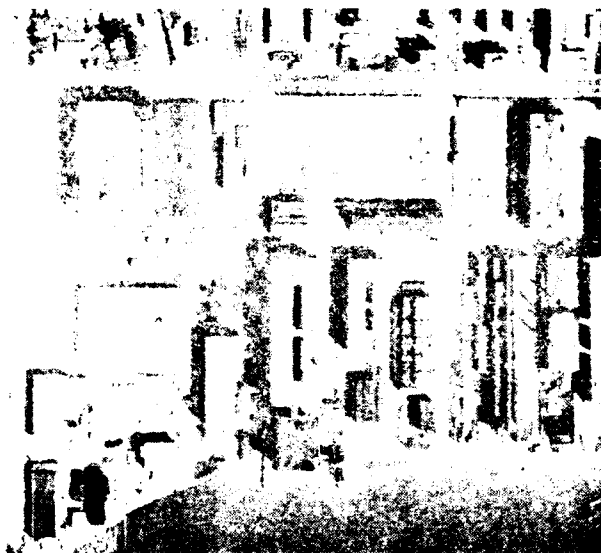


Figure 9: DC38008 Intensity Image.



Figure 10: Refined S2 Stereo Result.



Figure 11: Fine Intensity Segmentation.



Figure 12: Building Hypotheses.

In order to remove some or all of the false positives, we apply heuristics and constraints derived from monocular cues.

- Clusters of regions whose collective size is small or large with respect to possible building size are discarded.
- Regions that exist in areas marked as shadows [Irvin and McKeown 89, Shufelt and McKeown 93] are removed.
- Multi-spectral classification of regions may be used to remove elevated non-structural regions such as tree canopies [Ford and McKeown 92a].

Finally, the remaining clusters of buildings are re-analyzed by searching for a best fit building model for each cluster. In addition, the shadow regions

are again used to hypothesize the location of the shadow casting sides of each potential building, acting as a final cluster hypothesis verification.

3.1. Extraction test results

Some initial results are shown for a complex industrial scene, DC38008. Figure 9 shows the original intensity image of the left view of the stereo pair of aerial images, while Figure 10 shows the refined stereo result produced by the S2 matcher and used as the initial input to the building extraction process.

Figure 11 shows the segmentation of the scene in Figure 9 that was generated during the stereo refinement process in which the range of intensity within each region is ± 5 .

These regions are used as an initial over-segmented view of the aerial scene and adjacent regions are merged using the mode stereo disparity value within the region. The threshold for merger of adjacent regions is ± 1 pixel. Next individual regions are marked as potential buildings based on their relationship to adjacent regions. According to the heuristic rule: (1) If the mode disparity within the region is less than the lowest of its neighbors, then it is not considered a building; (2) If its disparity is greater than the highest of its neighbors, then it is given a likelihood value of 1.0 (from a range of 0-1.6); (3) If the mode disparity is equal to both the high and low values of its neighbors, it is allowed to be considered a building hypothesis and assigned a value of 1.6; Otherwise, its likelihood is calculated by the formula:

$$\frac{1.5 \times D - L - 0.5 \times M}{H - L}$$

where:

- D is the disparity of the local region,
- M is the mode of the surrounding regions,
- H is the maximum disparity of the surrounding regions, and
- L is the minimum disparity of the surrounding regions

Figure 12 shows the result of accepting regions rated at 0.5 or according to the above heuristic. Clusters of regions that are very large (more than 6000 pixels) or small (less than 100 pixels) are removed.² This is followed by a further restriction that all clusters that do not have a hypothesized shadow region to their non-sunward edges are removed. Figure 13 shows the final result after these restrictions.

As a result of this process many of the significant buildings are detected, with various degrees of accurate delineation. One way to visualize the results is to look at the differences between a three-dimensional ground truth description, the refined stereo disparity map, and the three-dimensional scene that results from using building extraction. Figure 14 shows the original scene rendered using a hand-generated stereo ground truth estimate 14(a), the shows the refined S2 stereo result 14(b), and using the building hypotheses generated by this technique 14(c).

² Here a "cluster" is defined as the transitive closure of adjacent (within 2 pixels) regions.

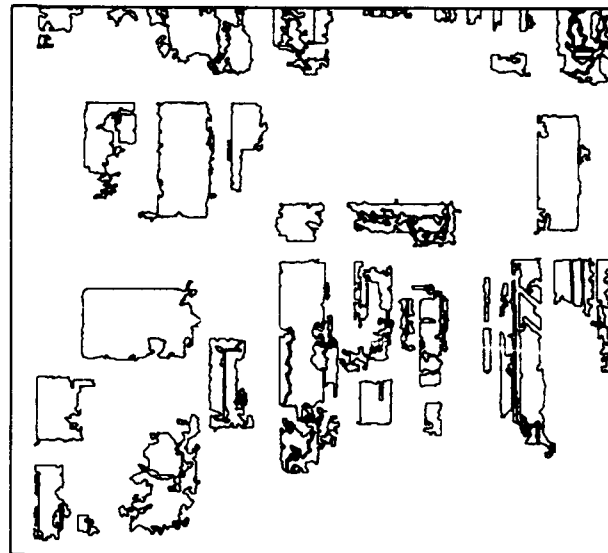


Figure 13: Filtered Building Hypotheses.

3.2. Open issues

Although our initial results are promising, we feel that no approach will reliably detect and delineate manmade structures solely by using stereo disparity. As a part of the cooperative-methods paradigm we plan to include other sources of information such as BABE building hypotheses [McKeown 90] and surface material classification [Ford and McKeown 92b, Ford and McKeown 92a]. In rugged terrain or in areas with significant tree canopy additional cues will be necessary for both the selection and the filtering of building hypotheses. In addition, we expect that such monocular cues, such as those generated by BABE will play an important role in the verification and re-analysis of region clusters during the model fitting and labeling process.

4. Manual selection of building hypotheses

Automated feature extraction from aerial images is a complex problem, and research in this domain has illustrated the difficulties in reliably detecting and verifying building structure. Although the ultimate goals of our work in this area are systems which will accurately detect and precisely delineate man-made features in aerial photography without human intervention, it is clear that a combination of current extraction techniques with some degree of user guidance has the potential to exhibit improved performance on complex imagery.

To date, many of the semi-automated systems require a large portion of the detection and delineation tasks to be performed by the user of the system. In such systems, the user interactively manipulates a variety of models over features in the image, fitting the models to the features



(a) DC38008 Ground Truth



(b) Refined S2 Stereo Results



(c) Visualization after Building Extraction

Figure 14: DC38008 Building Extraction.

[Hanson et al. 87, Kass et al. 87]. An alternative paradigm suggests that another approach for developing a high-performance system is to allow the user to lend a guiding hand during the execution of the feature extraction algorithms.

4.1. Manual selection of building hypotheses

We have been exploring possibilities for the application of human interaction in the extraction process. Our current testbed for this research is BABE, a line-corner intensity based feature extractor [McKeown 90]. In brief, BABE proceeds through four major phases to incrementally generate building hypotheses. The first phase constructs corners from lines, under the assumption that buildings can be modeled by straight line segments linked by (nearly) right-angled corners. The second phase constructs chains of edges which are linked by corners, to serve as partial structural hypotheses. The third phase uses these line-corner structures to hypothesize boxes, parallelopipeds which may delineate man-made features in the scene. The fourth phase evaluates the boxes in terms of size and line intensity constraints, and the best boxes for each chain are kept, subject to shadow intensity constraints similar to those proposed by [Nicolin and Gabler 87] and [Huertas and Nevatia 88].

In recent work, we have addressed the possibility of replacing the hypothesis evaluation routine with a simple form of user verification, in which a person uses the mouse to drop points on each individual structure in the scene. Then, hypothesis evaluation reduces to determining which boxes produced by BABE contain points placed by the user. This level of interaction does not place great demands on the user, and makes effective use of the hypothesis generation capabilities of BABE; thus, it serves as an interesting test for an intermediate level of man-machine interaction in this domain.

Figure 15 shows a ground-truth hand segmentation of a suburban scene in Washington, DC. Figure 16 shows the complete set of hypotheses generated by BABE for this scene, and Figure 17 shows the hypotheses verified by the shadow intensity constraint algorithms invoked in the fully automatic version of BABE. Figure 18 illustrates the results of a semi-automated BABE execution, in which the shadow verification algorithm was replaced by user selection of three points on each building, followed by intersection of these points with the full set of hypotheses in Figure 16. Each of these results was then compared on a pixel-by-pixel basis with the ground-truth hand segmentation to generate the statistics in Table 2. Note that we give data for a single-point user selection example as well; we omit the corresponding figure for brevity.

Results	Building Detection	Background Detection	Total Detection
All Boxes	82.1%	63.0%	65.8%
Best Boxes	57.4	97.7	91.9
1 Point	67.7	94.1	90.2
3 Point	77.1	92.9	90.6

Table 2: Building/background detection statistics

With the simple mechanism of multiple point selection, a user interacting with BABE can achieve a marked improvement in building detection, at the slight expense of accumulating errors in background classification. This is due to line placement errors in BABE hypotheses that are otherwise accurate descriptions of man-made structure. Note also, however, that the total scene classification rate remains essentially the same in each of the three examples. This suggests that user interaction at the verification level trades detection rate against overall classification precision.

4.2. Open issues

Given that the initial hypothesis data produced by BABE still fails to detect 18% of the building structures in the scene, it should be clear that more work is necessary on the basic feature extraction algorithms, and we intend to continue our pursuits in this area. User interaction at an intermediate level appears to be a fruitful avenue for further exploration, however, and we intend to investigate this topic further. One key issue is the determination of the appropriate level of interaction between a user and a feature extraction algorithm. We also plan to experiment with user input at other phases in the extraction algorithms, such as corner detection, line linking, and structure generation.

5. Terrain Modeling and Visualization

Terrain modeling is becoming an increasingly important issue with the advent of large-scale distributed simulations for training, mission rehearsal, and mission planning. Such systems rely on efficient representations for natural terrain, as well as manmade features such as buildings, roads, and bridges. Our recent work in this area has focused on the development of visualization tools for three-dimensional data, and in the continuation of our research in triangular irregular networks (TINs).

5.1. Visualization tools

With the increasing availability of a variety of digital spatial data ranging from map databases, object model descriptions, digital elevation

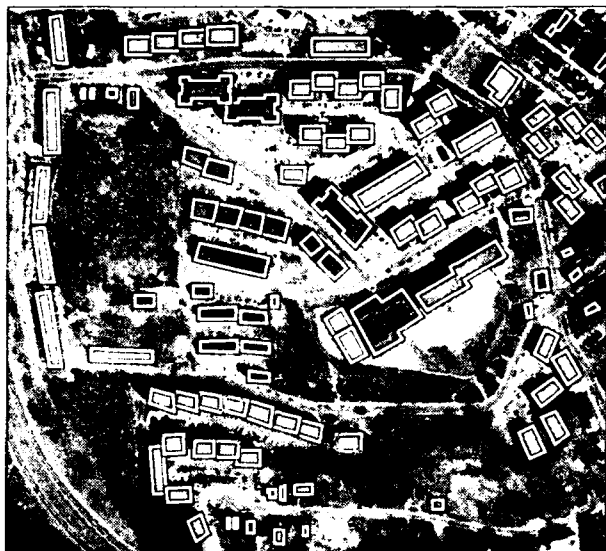


Figure 15: Ground truth segmentation for DC37405.

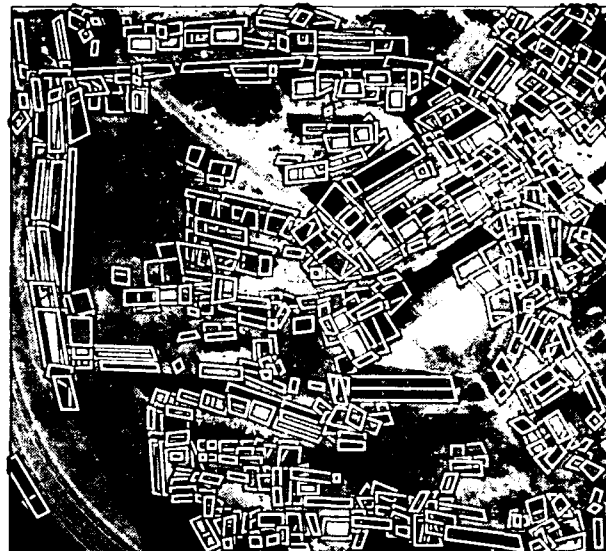


Figure 16: Hypothesis generation for DC37405.



Figure 17: Automatic verification for DC37405.

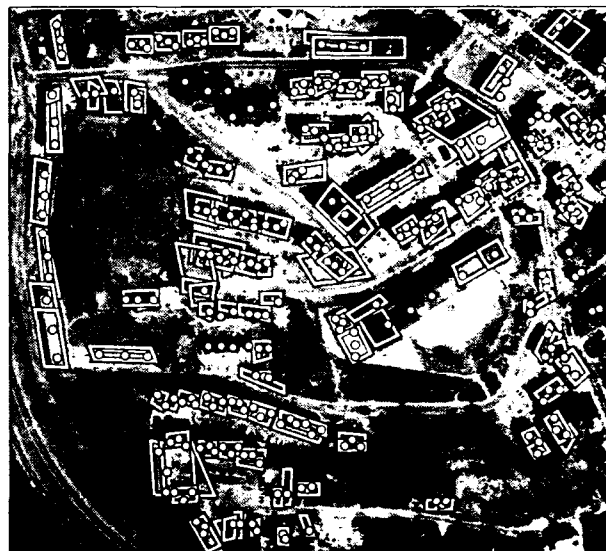


Figure 18: User-assisted 3 point verification.

models, and geo-referenced imagery, there is a need to conveniently view image and vector data to support various aspects of our research. In many cases these datasets are best visualized in three dimensions. Figure 19 demonstrates the difference between viewing terrain as an intensity mapped height field and as an overhead shaded relief rendering. The latter process takes into account shading from a light source and tends to make the surface structure more apparent. Small changes in terrain detail are enhanced and surface slope and aspect appear more pronounced. To support our need for 3D display of spatial data we have developed an X/Motif application, XRELIEF, to allow us to visualize digital elevation models and TINs, manual ground truth segmentations,

automated stereo results, multi-spectral results, and digital map data (ITD, DLMS) overlaid on terrain.

Figure 20 shows a sample control panel used to specify imagery, terrain, map overlay, and viewing parameters. Users can create and store multiple ordered sets of camera parameters in order to compare results from different stages of an extraction process. They can also compare results from different analysis methods from a single known viewpoint. XRELIEF has an intuitive graphical interface for control and creation of these camera parameters, as well as positioning of an illumination source used for shading calculations, and simple animation support. This interface is shown at the lower right of Figure 20. The large

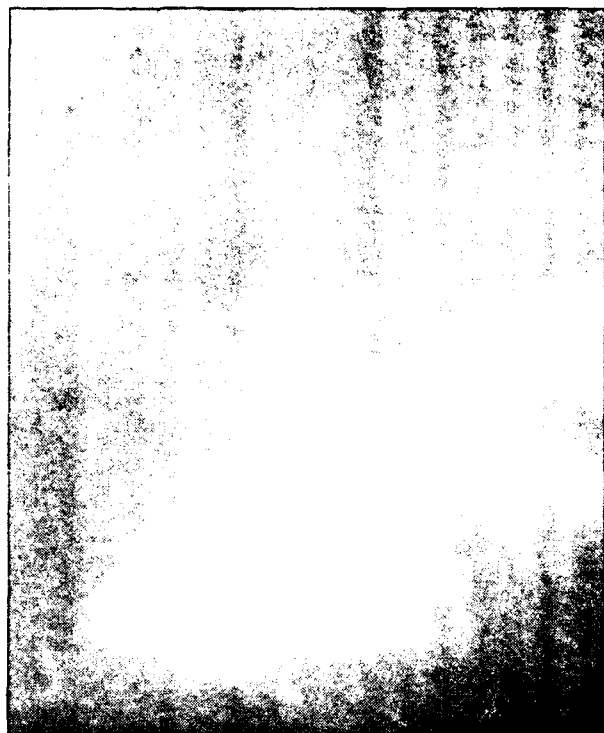


Figure 19: Comparison of 2D and 3D display of Digital Elevation Models.

and small circles represent camera lookfrom and lookat points, and the image displayed underneath corresponds to the image being overlaid on the terrain.

5.2. A new TIN generation method

A digital elevation model (DEM) is a terrain model consisting of elevation data regularly spaced on a grid. A triangular irregular network consists of elevation data that are irregularly spaced and are connected into triangular facets to form a surface. The ability of the TIN to place points irregularly permits point density to adapt to terrain complexity, and allows points to be placed precisely on peaks and valley floors. The TIN terrain model is ideally suited to real-time rendering, as it consists of a reduced set of polygons tailored to the underlying terrain complexity. Previous research in TIN generation using point selection from the DEM was described in an earlier paper [Polis and McKeown 92]. In this section we give a brief update on our development of a new (and improved) method for point selection.

Our point selection method relies on the iterative selection of points based upon successive approximation to the actual terrain surface. At each iteration a dense DEM is constructed by interpolation from the current TIN. This

approximate TIN is compared to the actual DEM and the point or points having the greatest error in elevation are determined. These DEM points are added to the TIN as correction points, and a new triangulation is generated. The triangulation is evaluated based on a global point budget and the residual global error. If the point budget is not exceeded and the RMS error is still greater than a user specified error goal, then the process is repeated. In practice the point budget controls the stopping conditions for the TIN generation process.

Our previous point selection method relied on the generation of error contours and associated medial axes. Points would be selected from the set of maximal contours generated at each iteration. The new point selection is based solely upon a measure calculated at each point in the approximation DEM. The new process chooses fewer correction points per iteration and as a result many more iterations are required. However, the points chosen are of higher quality in terms of our RMS error metric. As a result of this a fast triangulation is necessary, so the modified greedy triangulation has been replaced with a Delaunay triangulation. However, the improvement in point selection appears to outweigh the loss in triangulation accuracy, especially since the iterative process will naturally add points to correct poor triangulations.

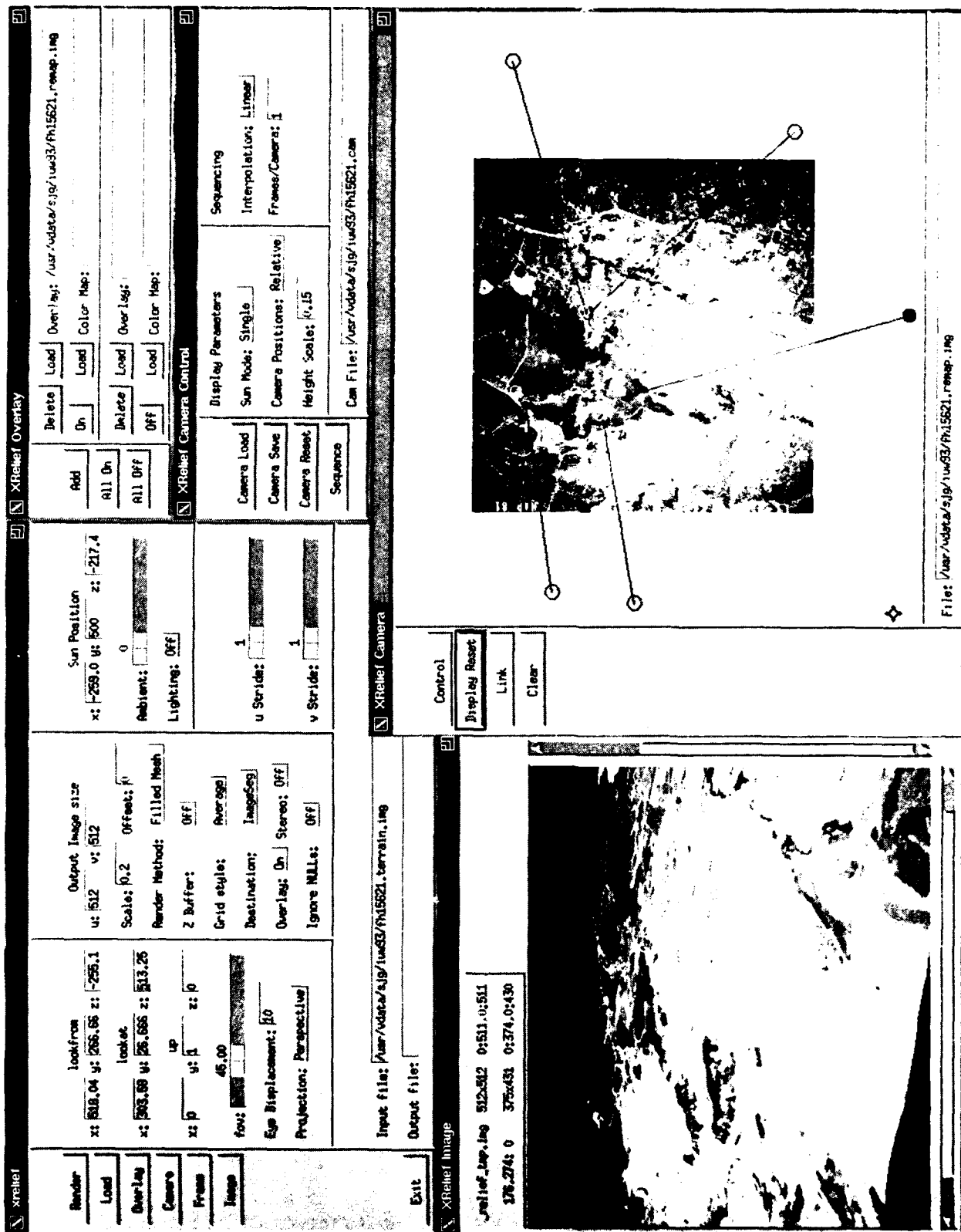


Figure 20: User control panel for XRELIEF.

In the following section we will describe the use of this new triangulation method to generate large-scale TINs suitable for use in SIMNET.

5.2.1. TIN generation for SIMNET

A digital elevation model constructed to support a SIMNET training exercise was provided to us by the U.S. Army Topographic Engineering Center (USATEC). The DEM covered an area 50 kilometers on a side (2500 square km) including the National Training Center (NTC), Fort Irwin, California. This area is primarily desert, with some highly eroded mountainous areas and intricate alluvial fans running to the desert floor. The sheer size of the area presents significant problems. The DEM consists of 1979x1979 points, nearly 4 million elevation posts. To maintain the desired polygon density for the SIMNET computer image generation systems, only 90,000 points were to be selected for the TIN.

An additional complication for the TIN generation process was the desire for reduced fidelity in the mountainous areas with increased detail in the areas of alluvial fans and on the desert floor. This was primarily driven by the fact that mountainous areas are not accessible to ground vehicles (simulated or otherwise) yet, due to their height and complexity, they tend to accumulate a large number of TIN points. This decreases the budget available for other areas of the terrain. An overlay indicating the mountainous areas was provided by USATEC, and was used to produce an importance grid. Our initial experiment was to make the maximum error in the mountains approximately one fifth as large as that in the low lying areas. We smoothed the importance grid to avoid problems that might result from a discontinuity at the boundary of the mountainous area. Since point selection under our new method was based solely upon a measure calculated at each point, it was now possible to use the importance grid to apply a weight to each point based upon its subjective importance.

Figure 21 shows a shaded relief representation of the western part of the National Training Center. The left half shows the terrain relief using the original digital elevation model. The right half shows the same area using the TIN representation for the underlying surface structure. The TIN was generated using selective fidelity in the mountainous areas. Using approximately 2.5% of the original DEM points we were able to construct a TIN with an RMS elevation error of 3.1 meters when compared to the original DEM. The range of elevations in the DEM was approximately 1500 meters. From a qualitative standpoint it appears that the major topographic features are generally preserved and that detail in the alluvial fans and desert floor areas are also quite good. This

impression was confirmed using the SIMNET system at USATEC and driving an M1 tank (simulated) through the terrain.

5.3. Open issues

We have shown the utility of our new TIN construction method for a large-scale digital elevation model. Research issues remain in determining how to factor more detailed mobility information into the point selection process. We are also interested in addressing how to integrate small scale cartographic features, particularly roads into a TIN, while maintaining a limited polygon budget.

From a pragmatic standpoint, the generation of a TIN directly from the NTC digital elevation model using our new point selection method would take weeks, even on a fast (20mips) workstation. Our initial solution was to divide the DEM into tiles and then generate a TIN for each tile. We maintained a restriction that TINs must match along common boundaries. The execution time is divided by the number of tiles, and can be further reduced since tiles which have no common boundary can be generated in parallel. Using this method we were able to generate the NTC TIN overnight using three workstations. There are limits to this technique since as the number of tiles is increased, the global behavior of point selection is greatly reduced. This can defeat the overall goal of placing points wherever their utility is the greatest.

6. Toward Knowledge Refinement for Large Rule-Based Systems

Knowledge refinement is a central problem in the field of expert systems [Buchanan and Shortliffe 84]. It refers to the progressive refinement of the initial knowledge-base of an expert system into a high-performance knowledge-base. For rule-based systems, refinement implies the addition, deletion and modification of rules in the system so as to improve the system's *empirical adequacy*, i.e., its ability to reach correct conclusions in the problems it is intended to solve [Ginsberg et al. 88].

The goal of our research effort is to understand the methodology for refining large rule-based systems, as well as to develop tools that will be useful in refining such systems. The vehicle for our investigation is SPAM, a production system (rule-based system) for the interpretation of aerial imagery [McKeown et al. 89, McKeown et al. 85]. It is a mature research system having over 600 productions, many of which interact with complex geometric algorithms. A typical scene analysis task requires between 50,000 to 400,000 production firings and an execution time of the

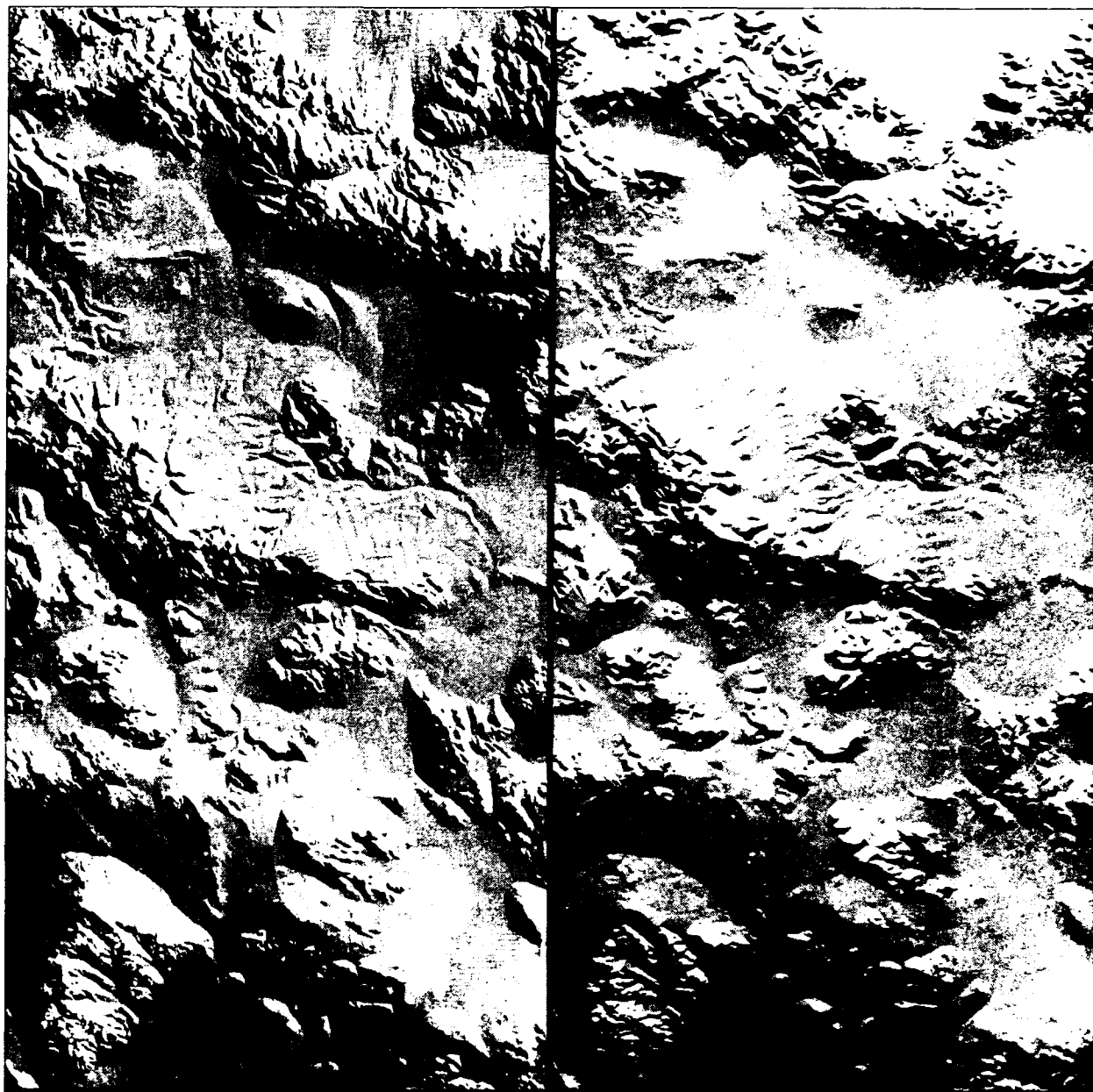


Figure 21: SIMNET 50km NTC DEM (left) juxtaposed with 50km NTC TIN (right)

order of 2 to 4 cpu hours.³

Large, compute-intensive systems like SPAM impose some unique constraints on knowledge refinement. First, the problem of credit/blame-assignment is complicated. It is extremely difficult to isolate a single culprit production (or a set of culprit productions) to blame for an error observed in the output. Second, given the large run-time, it is not possible to rely on extensive experimentation for knowledge refinement.

³ Using a highly optimized C based OPS5 [Kalp et al 88] running on a DEC 5000/200

As a result, the methodology adopted in well-known systems such as SEEK and SEEK2 [Politakis and Weiss 84, Ginsberg et al. 88], or KRUST [Craw and Sleeman 91], cannot be directly employed to refine knowledge in SPAM. Our approach is to address this problem in a bottom-up fashion, i.e., begin by understanding SPAM's individual phases, and then attempt to understand the interactions between the phases. A different set of tools is required to allow the user to focus attention on individual modules responsible for intermediate results and refine them. In our work so far, we have focused on the second phase in SPAM, local-consistency (LCC), which applies constraints to a

set of plausible hypotheses and prunes the hypotheses that are inconsistent with those constraints. Furthermore, we have narrowed this focus to refining SPAM's distance and orientation constraints.

In working toward refining these constraints, we posed several questions to help guide our analysis:

1. Does this constraint play a positive (helpful) or negative (unhelpful) role in the interpretation process?
2. If the role is positive, are there improved constraint bounds values?
3. What is this constraint's impact on run time?
4. Is this constraint uniformly applicable or should it be applied selectively? If selectively, in what cases should we apply the constraint?

In the following sections we describe some of our current efforts toward addressing these questions.

6.1. The refinement methodology

We have begun our investigation on knowledge refinement by focusing on SPAM's second phase of processing, LCC. This phase was chosen because most of SPAM's time is spent in this phase, and it showed the most potential for future growth. LCC performs a modified constraint satisfaction between hypotheses generated in SPAM's first phase. In LCC, a successful application of a constraint provides support for a pair of hypotheses, and an unsuccessful application goes towards filtering out that pair of hypotheses. The distance constraint specifies allowable distance ranges between different pairs of hypothesized objects, e.g., two hangar buildings must occur between 20 and 200 meters apart, while a parking apron and a hangar building must be between 0 and 50 meters apart. In essence, each constraint in the LCC phase classifies the pairs of hypotheses --- either the constraint supports that pair, or it does not.

Our refinement methodology consists of three parts: intermediate result evaluation, constraint optimization, and embedded evaluation. The first two methods allow the isolation and improvement of individual constraints, while the third method allow us to evaluate the performance of the new knowledge in the context of the overall system output.

6.1.1. Intermediate result evaluation

In order to measure the effect of various spatial constraints we needed to establish a database of correct inputs and outputs for LCC. For each of the sets of data that we run through SPAM we have a ground-truth database containing all the objects in

the scene with their correct hypothesis labels. An "ideal" input to the LCC phase, a set of hypotheses that are 100% correct, can easily be manually generated and run through the system. Any errors in the output are then directly attributable to the constraints.

A set of constraint results can be generated by allowing a user (the *expert*) to enumerate those constraints that should exist between each pair of objects in the ideal input. This is equivalent to an "ideal" output for LCC.

Once SPAM has processed the ideal input, the generated output can then be compared to the ideal output. Such a comparison is informative as it allows a quantitative measure of error to be computed. We can produce this comparison as a set of confusion matrices where each matrix represents the results for a single constraint and a single pair of classes. These matrices contain the usual cells (true-positives, false-positives, true-negatives, false-negatives). An example confusion matrix is shown in Figure 22.

A true-positive entry in the confusion matrix indicates situations where the expert and LCC both conclude that the constraint supports a pair of hypotheses. A true-negative entry indicates situations where the expert and LCC both conclude that the constraint does not support a pair of hypotheses. A false-positive entry is one where LCC concludes support, while the expert does not. A false-negative entry is one where the expert concludes support, while LCC does not.

6.1.2. Constraint optimization

By examining the overlap of each histogram, we can tell if SPAM's distance constraint is working properly. The overlap of, for example, true positives with false positives can tell us how the constraint can be modified to achieve the greatest number of true positives without introducing too many false positives. For numeric constraints, such as distance, we have developed an automatic process for adjusting the constraint bounds to generate an improved set of ranges.

Automated bounds selection is achieved by doing an exhaustive search through the space of possible bounds settings, evaluating each setting with an objective function. Currently, this objective function weighs all cells in the confusion matrix equally and seeks to maximize the number of elements in the diagonal cells of the matrix (true-positives, true-negatives).

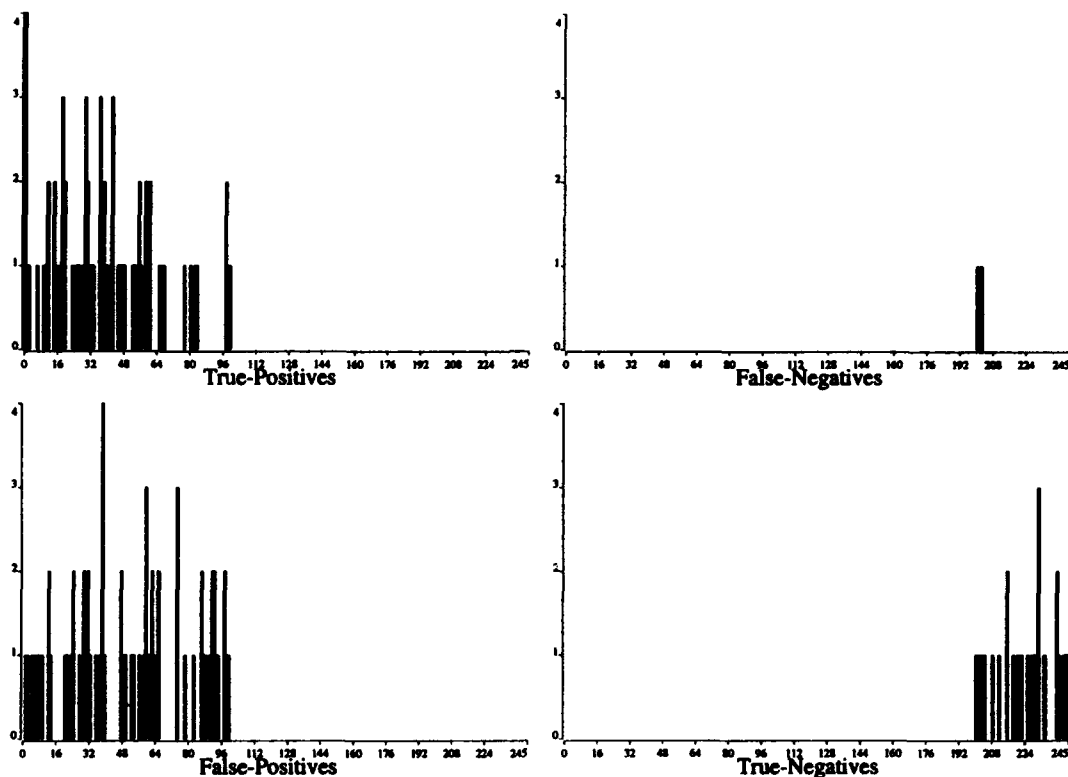


Figure 22: Confusion matrix showing correctness of SPAM's hangar-building/road distance constraint.

6.1.3. Embedded evaluation

Both methods described above evaluate and improve the performance of isolated constraints. However, it is most important that the system's overall output improve with the adjusted constraint embedded within it.

We want to choose to evaluate embedded performance at a place within the system where the intermediate results have been used, but where only a small amount of processing has been done so that the credit assignment problem is avoided. We chose to evaluate performance at the end of SPAM's third phase, FA. This phase does grouping based on the results of the constraints applied in LCC. These groups of supporting hypotheses are called *functional-areas* (FAs).

SPAM's long run times prohibit iterative refinement if the number of iterations required can be large. This limitation can be avoided by appropriately choosing experiments to run and observing the system's behavior. In this way, we sample the space of possible bounds settings and hence, sample the system's output behavior.

6.2. Analysis of results

We ran the LCC phase of SPAM with a set of hand-labeled hypotheses and compared this to our

ground-truth. The resulting comparison histogram was used by our bounds adjustment procedure which generated a set of optimal settings for this constraint. These experiments were performed on four data sets, with each data set corresponding to a different airport scene.

Next, we ran five experiments for each data set, allowing SPAM to execute through it's third phase. Each experiment corresponded to a modification of the distance constraint bounds, as follows:

<i>off</i>	constraint disabled;
<i>low</i>	bounds set to minimum value;
<i>optimized</i>	optimal bounds setting arrived at by adjustment procedure;
<i>original</i>	original bounds setting optimization process;
<i>high</i>	bounds set to maximum value;

Those table entries labeled *orient-** are orientation constraint modifications. For each run, we compiled statistics on run-time, number of production firings, number of functional-areas generated, and number of correct and incorrect hypotheses included in those functional-areas. Evaluation was done by comparing each run to the *original* bounds settings.

Dataset	Bounds Setting	Time (cpu hours)	No. Rule Firings	No. of FAs	No. of Correct (diff.)	No. of Incorrect (diff.)
Moffett	off	0.26	97051	16	-21	-19
"	low	0.29	103779	16	-21	-19
"	optimized	0.30	113107	16	-10	-6
"	original	0.30	115012	17	0	0
"	high	0.35	135978	19	16	55
"	orient-off	0.27	104672	17	-1	-21
"	orient-low	0.30	114783	18	-1	2
"	orient-high	0.32	124697	18	1	8
DC National	off	0.52	161685	27	-25	-82
"	low	0.57	170801	27	-26	-83
"	optimized	0.57	180060	28	-9	-44
"	original	0.58	183273	28	0	0
"	high	0.72	240726	28	68	210
"	orient-off	0.54	167618	28	-16	-18
"	orient-low	0.57	186919	28	3	6
"	orient-high	0.62	203931	28	27	14

Table 3: Sampled SPAM performance, measured along several dimensions while changing constraint settings.

Our results are presented in Table 3. From this table, we can make several observations. First, from the increase in run time (from *off* to *original*), it can be noted that the distance constraint is having some impact on the results. The increase in the number of correct hypotheses and the drop in the number of incorrects reveals that this constraint is playing a positive role.

Finding the best setting for the bounds of the constraints is a more difficult problem. The evaluation function for this task seems very complex, taking into account relationships between numbers of corrects/incorrects, sizes of functional-areas, and run time. For the Moffett data set, the number of corrects increases, while the number of incorrects increases, but at a slower pace. From this we would conclude that the bounds should be set to the maximum value. However, the same analysis for DC National implies that the optimized value would be best. Other larger data sets, such as those for San Francisco National Airport, show a similar trend. This suggests that the bounds for the distance constraint should be chosen on a case by case basis.

An interesting phenomena is observed as the constraints are selectively turned off. The generated functional-area groups get smaller, but they do not radically change in area of coverage. This implies that the distance constraint is selectively applicable, i.e., it largely overlaps with the other system constraints, but it is necessary for the inclusion of some subset of hypotheses. Because optimizing and then coupling these two constraints does not produce a dramatic improvement in results, it appears that more constraints may be required to do a better job of interpretation.

6.3. Open issues

With the recent emphasis on performance evaluation of vision systems focused upon low and intermediate level vision tasks, this work establishes a data point in the area of high level vision systems. Though our goal is to improve the interpretations generated by SPAM, we have begun by improving our understanding of how the individual components of SPAM operate, and how they interact. This will provide the foundation for

understanding the effects of modifying or adding knowledge to the system.

We have been able to show that SPAM's distance constraint plays a positive role in the interpretation task. However, choosing an "optimal" value is difficult, and seems to be scene dependent. Finally, we have determined that, of the two constraints considered thus far (distance and orientation), the applicability of both overlaps a great deal.

There is still much to be done. In the short term, there are several obvious problems that we have not addressed. First, we need to look more closely at the applicability of the constraints and characterize, if possible, the cases where each constraint can be applied. Second, it is unclear if the bounds optimization procedure we developed will extend to non-numeric constraints. Finally, we wish to extend the analysis to simultaneously perform validation across multiple constraints.

Overall, we believe that it will be possible to build a heuristic system that would automate the knowledge refinement process, similar to the automated system in [Ginsberg et al. 88, Politakis and Weiss 84]. Within such a system, we would like to discover ways not only to improve the current constraints, but to automate methods for determining what new knowledge may be needed.

Our work in multispectral analysis to determine surface material properties has been focused on basic research on demonstrating the utility of such data for cartographic feature extraction. For many tasks in traditional remote sensing it is clear that having surface material information drives many tasks in land use, environmental monitoring, and natural resource management. Our hypothesis is that such data can aid in manmade object detection, delineation, and identification. However, getting multispectral imagery at spatial resolutions that are comparable with the high resolution panchromatic imagery has been difficult.

Initial work has demonstrated the utility of the refinement of multispectral classification using monocular panchromatic imagery, and the fusion of stereo disparity maps with surface material information [Ford and McKeown 92b, Ford and McKeown 92a]. One issue is maintaining accurate registration between the multispectral scanner data (8 meter gsd) and the panchromatic imagery (1.3 meter gsd). Once this is accomplished a unique hybrid three dimensional multispectral dataset can be created and utilized for further analysis.

Our recent research has been to perform a performance evaluation of two classification

techniques, gaussian maximum likelihood and differential radial basis function, for surface material classification. In order to do this evaluation we have created several highly detailed ground truth segmentations based upon manual analysis of the multispectral imagery, as well as by inspection of panchromatic imagery acquired over the same area. Details of this work can be found in a companion paper [Ford et al. 93] in this volume.

Our overall conclusions are that multispectral imagery with moderate spatial resolution has great potential to provide scene domain cues necessary to improve the performance of cartographic feature extraction based on panchromatic imagery with high spatial resolution.

7. Multi-spectral classification

Our work in multispectral analysis to determine surface material properties has been focused on basic research on demonstrating the utility of such data for cartographic feature extraction. For many tasks in traditional remote sensing it is clear that having surface material information drives many tasks in land use, environmental monitoring, and natural resource management. Our hypothesis is that such data can aid in manmade object detection, delineation, and identification. However, getting multispectral imagery at spatial resolutions that are comparable with the high resolution panchromatic imagery has been difficult.

Initial work has demonstrated the utility of the refinement of multispectral classification using monocular panchromatic imagery, and the fusion of stereo disparity maps with surface material information [Ford and McKeown 92b, Ford and McKeown 92a]. One issue is maintaining accurate registration between the multispectral scanner data (8 meter gsd) and the panchromatic imagery (1.3 meter gsd). Once this is accomplished a unique hybrid three dimensional multispectral dataset can be created and utilized for further analysis.

Our recent research has been to perform a performance evaluation of two classification techniques, gaussian maximum likelihood and differential radial basis function, for surface material classification. In order to do this evaluation we have created several highly detailed ground truth segmentations based upon manual analysis of the multispectral imagery, as well as by inspection of panchromatic imagery acquired over the same area. Details of this work can be found in a companion paper [Ford et al. 93] in this volume.

Our overall conclusions are that multispectral imagery with moderate spatial resolution has great potential to provide scene domain cues necessary to improve the performance of cartographic feature

extraction based on panchromatic imagery with high spatial resolution.

Acknowledgments

We thank the unsung hackers of the Digital Mapping Laboratory for their help in the research reported in this paper. Steve Lacy pointed the way toward user assisted building verification, Chris Olson motivated his way through visualization routines, Jeff McMahill performed multispectral magic, and Scott Colville knows more than he'd like about functional areas. Ed Allard, Karl Fischer, and Mark Stemm joined the project too recently to have done anything terribly interesting.

References

- [Buchanan and Shortliffe 84]
Bruce G. Buchanan and Edward H. Shortliffe. *The Addison-Wesley series in artificial intelligence: Rule-based expert systems: (The MYCIN experiments of the Stanford Heuristic Programming Project.)* Addison-Wesley, Reading, Massachusetts, 1984.
- [Craw and Sleeman 91]
S. Craw and D. Sleeman. The Flexibility of Speculative Refinement. In *Proceedings of the Eighth International Workshop on Machine Learning (ML91)*, pages 28-32. 1991.
- [Ford and McKeown 92a]
S. J. Ford and D. M. McKeown. Utilization of Multispectral Imagery for Cartographic Feature Extraction. In *Proceedings of the DARPA Image Understanding Workshop*, pages 805-820. Morgan Kaufmann Publishers, Inc., San Mateo, CA, January, 1992.
- [Ford and McKeown 92b]
S. J. Ford and D. M. McKeown. Information Fusion of Multispectral Imagery for Cartographic Feature Extraction. In *Commission VII: Interpretation of Photographic and Remote Sensing Data*. Washington, DC, XVII ISPRS Congress, Aug. 2-14, 1992.
- [Ford et al. 93]
S. J. Ford, J. B. Hampshire and D. M. McKeown. Performance Evaluation of Multispectral Analysis for Surface Material Classification. In *Proceedings of the DARPA Image Understanding Workshop*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, April, 1993.
- [Ginsberg et al. 88]
A. Ginsberg, S. Weiss and P. Poliatkis. Automatic knowledge-base refinement for classification systems. *Artificial Intelligence* 35:197-226, 1988.
- [Hanson et al. 87]
A. J. Hanson, A. P. Pentland and L. H. Quam. Design of a Prototype Interactive Cartographic Display and Analysis Environment. In *Proceedings: DARPA Image Understanding Workshop*, pages 475-482. February, 1987.
- [Hsieh et al. 92]
Y. C. Hsieh, D. M. McKeown and F. P. Perlant. Performance Evaluation of Scene Registration and Stereo Matching for Cartographic Feature Extraction. *Pattern Analysis and Machine Intelligence* 14(2):214-238, February, 1992.
- [Huertas and Nevatia 88]
A. Huertas and R. Nevatia. Detecting buildings in aerial images. *Computer Vision, Graphics and Image Processing* 41:131-152, April, 1988.
- [Irvin and McKeown 89]
R. B. Irvin and D. M. McKeown. Methods for exploiting the relationship between buildings and their shadows in aerial imagery. *IEEE Transactions on Systems, Man and Cybernetics* 19(6):1564-1575, November, 1989.
- [Kalp et al. 88]
D. Kalp, M. Tambe, A. Gupta, C. Forgy, A. Newell, A. Acharya, B. Milnes and K. Swedlow. *Parallel OPS5 User's Manual*. Technical Report CMU-CS-88-187, Computer Science Department, Carnegie Mellon University, November, 1988.
- [Kass et al. 87]
M. Kass, A. Witkin and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision* 1(4):321-331, 1987.
- [McGlone and Shufelt 93]
J. Chris McGlone and Jefferey A. Shufelt. Incorporating vanishing point geometry into a building extraction system. In *Proceedings of the Image Understanding Workshop*. Morgan Kaufmann Publishers, Inc., April, 1993.
- [McKeown 87]
D. M. McKeown. The Role of Artificial Intelligence in the Integration of Remotely Sensed Data with Geographic Information Systems. *IEEE Transactions on Geoscience and Remote Sensing* GE-25(3):330-348, May, 1987.
- [McKeown 90]
D. M. McKeown. Toward automatic cartographic feature extraction. In Pau, L. F. (editor), *NATO ASI Series. Volume F 65: Mapping and Spatial Modelling for Navigation*, pages 149-180. Springer-Verlag, Berlin Heidelberg, 1990.
- [McKeown and Hsieh 92]
David M. McKeown and Yuan C. Hieh. Hierarchical Waveform Matching: {A New Feature-Based Stereo Technique.} In *Computer Vision and Pattern Recognition*, pages 513-519. Champaign, Illinois, June 15-18, 1992.

[McKeown and Perlant 92]

D. McKeown and F. Perlant. Refinement of Disparity Estimates Through the Fusion of Monocular Image Segmentations. In *Computer Vision and Pattern Recognition*, pages 486-492. Champaign, Illinois, June 15-18, 1992.

[McKeown et al. 85]

D. M. McKeown, W. A. Harvey and J. McDermott. Rule Based Interpretation of Aerial Imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-7(5):570-585, September, 1985.

[McKeown et al. 89]

D. M. McKeown, W. A. Harvey and L. Wixson. Automating Knowledge Acquisition For Aerial Image Interpretation. *Computer Vision, Graphics and Image Processing* 46(1):37-81, April, 1989.

[Nicolin and Gabler 87]

B. Nicolin and R. Gabler. A knowledge-based system for the analysis of aerial images. *IEEE Transactions on Geoscience and Remote Sensing* GE-25(3):317-329, May, 1987.

[Perlant and McKeown 90]

F. P. Perlant and D. M. McKeown. Scene Registration in Aerial Image Analysis. *Photogrammetric Engineering and Remote Sensing* 56(4):481-493, April, 1990.

[Polis and McKeown 92]

M. Polis and D. McKeown. Iterative TIN Generation from Digital Elevation Models. In *Proceedings of the DARPA Image Understanding Workshop*, pages 885-897. Morgan Kaufmann Publishers, Inc., San Mateo, CA, January, 1992.

[Politakis and Weiss 84]

P. Politakis and S. Weiss. Using empirical analysis to refine expert system knowledge bases. *Artificial Intelligence* 22:23-48, 1984.

[Shufelt and McKeown 93]

Jefferey A. Shufelt and David M. McKeown. Fusion of Monocular Cues to Detect Man-Made Structures in Aerial Imagery. *Computer Vision, Graphics and Image Processing: Image Understanding* 57(2), March, 1993.

Detection of Buildings from Monocular Views of Aerial Scenes Using Perceptual Grouping and Shadows

A. Huertas, C. Lin and R. Nevatia*

Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273

Abstract

We describe a system for detection and description of buildings in aerial scenes. This is a difficult task as the aerial images contain a variety of objects. Low-level segmentation processes give highly fragmented segments due to a number of reasons. We use a perceptual grouping approach to collect these fragments and discard those that come from other sources. We use shape properties of the buildings for this. We use shadows to verify the hypotheses generated by the grouping process. This step also provides 3-D descriptions of the buildings. Our system has been tested on a number of examples taken from the imagery supplied by the RADIUS program and the results have generally been very good. The current system is largely limited to overhead views, we are currently working on extensions to oblique views.

1 Introduction

The goal of this work is to detect and describe buildings from monocular views of aerial scenes. This is a difficult but important task for many applications such as photo-interpretation and cartography. There have been many previous attempts to solve this problem, in our group [Huertas, 1983, Huertas and Nevatia, 1988, Mohan and Nevatia, 1989] and elsewhere [Irvin and McKeown 1989, Liow and Pavlidis, 1990, Venkateswar and Chellappa, 1990]. These systems have shown interesting performance but on limited examples. The technique we describe in this paper, we believe, significantly extends the range of scenes that can be analyzed though many problems remain. We show several examples taken from the images provided by the RADIUS program to demonstrate the effectiveness of our technique.

Building detection is difficult for several reasons. The contrast between the roof of a building and surrounding structures such as curbs, parking lots, and walkways can be low. The contrast between the roofs of



Figure 1. A building from Ft. Hood, Texas

various wings, typically made of the same material, may be even lower. Low contrast alone is likely to cause low-level segmentation to be fragmented. In addition, small structures on the roof and objects, such as cars and trees adjacent to the building will cause further fragmentation and give rise to "noise" boundaries. Roofs may also have markings on them caused by dirt or variations in material. Shadows and other surface markings on the roof cause similar problems.

There are other characteristics of these images which may cause problems. Roofs have raised borders which sometimes cast shadows on the roof. This results in multiple close parallel edges along the roof boundaries and often these edges are broken and disjoint. At roof corners and at junctions of two roofs, multiple lines meet leading to a number of corners making it difficult to choose a corner for tracking. A roof cast a shadow along its side and often there are objects on the ground such as grass, trees, trucks, pavement, etc., which lead to changes in the contrast along the roof sides.

Consider the building in Figure 1, from a scene of Ft. Hood in Texas. The building is easy for humans to see and describe, but it is difficult for computer vision systems. Figure 2 shows the line segments detected in the image, using LINEAR, our linear feature extraction software [Nevatia and Babu, 1980, Canny, 1986]. We are still able to see the roof structures of the buildings readily and easily, but the complexity of the task now becomes more apparent. The building boundary is frag-

* This research was supported primarily by a subcontract from Hughes Aircraft Co. on the RADIUS program, and in part by the Defense Advanced Research Projects Agency under contract F49620-90-C-0078, monitored by the Air Force Office of Scientific Research. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

mented, there are gaps and missing segments. There are also many extraneous boundaries caused by other structures in the scene.

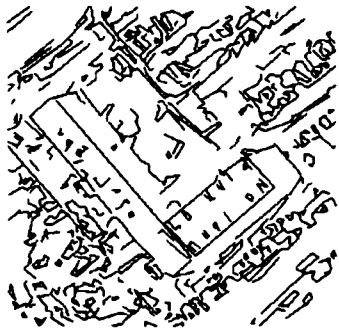


Figure 2. Line segments extracted from image

Much of the previous work to resolve these problems has used some form of a contour tracing technique, see for example [Huertas, 1983, Huertas and Nevatia, 1988, Venkateswar and Chellappa, 1990]. These are essentially local techniques that must make a decision of which path to trace at each local junction. Of course, all paths could be traced using backtracking but then the search space may become prohibitively large.

We propose, instead, to use a *perceptual grouping* approach. Cultural features such as buildings represent structures that are not random but have specific geometric properties. In this work, we restrict the shapes of buildings to be rectangular or composition of rectangular shapes (thus allowing L, T and I shapes for example). We also assume that the viewpoint is more or less overhead. Thus, primarily, we see roofs which project as rectangles or composition of rectangles. This property can be used to organize the detected line segments into roof hypotheses. We believe that this approach leads to many fewer hypotheses than would be generated by a complete contour tracing scheme.

We can choose among the many hypotheses by utilizing other properties of the image. Specifically, under favorable imaging conditions 3-D structures cast shadows that allow verification of roof hypotheses and further provide us an estimate of the height allowing us to generate 3-D descriptions of the detected buildings. An alternative would be to utilize more than one image of the scene to infer heights of the features of the roof and to separate them from features on the ground; another project in our group has explored this approach [Chung and Nevatia, 1992]. The advantage of using only one image, of course, is that such imagery is much easier to acquire.

Our approach combines several of the techniques from our previous work. Our perceptual grouping approach comes from the work described in [Mohan and Nevatia 1989], however, we use a very different selection technique. The earlier work, in fact, used perceptual grouping for stereo analysis, here we apply it to monocular analysis. Our shadow analysis method is an extension of the approach first described in [Huertas, 1983, Huertas and Nevatia, 1988].

2 Overview of the System

The diagram in Figure 3 shows the main components in our system. The system uses the line segments approximating the intensity boundaries to compute lines and relevant junctions among them. A hierarchy of features including parallel relationships and portions of rectangles leads to the formation of building hypotheses. These consist of instances of rectangular shapes that potentially correspond to building roofs and parts of building roofs (see section 3). Next, promising rectangles are selected and verified to correspond to building structures.

Our philosophy in the design of this system has been to make only those decisions that can be made confidently at each level. Thus, we choose to generate as many hypotheses as seem feasible at the first level. Our selection process too is conservative and favors keeping hypotheses that may be viable. The verification process has the most global information and can make stronger decisions. Even here, if our system is to be embedded in a larger system, some of the decisions would be deferred to that system where more context is available for decision making.

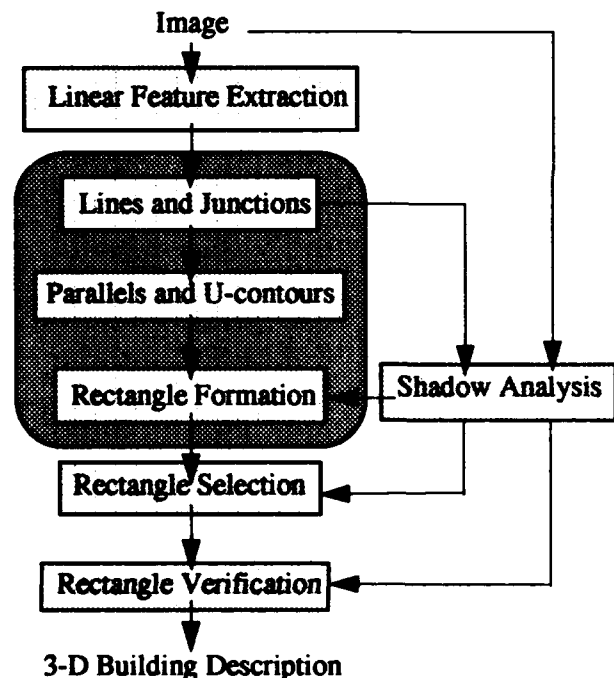


Figure 3. Block Diagram of the System

3 Generation of Hypotheses

The process of hypotheses formation is essentially the one described in [Mohan and Nevatia, 1989]. This system has been applied to building detection but using a stereo pair of images. In this process we construct a feature hierarchy which encodes the structural relationships specific to rectangular shapes: Lines, parallels, U-

contours, and rectangles.

Lines and Junctions

A group of close parallel lines represent a *linear structure* at a higher granularity level than the edges (see the common boundary between the building wings in Figure 2.) The resulting *lines* have a length and an orientation derived from the contributing elements. Figure 4 shows the lines obtained from grouping the segments in Figure 2. We use these lines to detect *L-junctions* and *T-junctions* also shown in Figure 4.

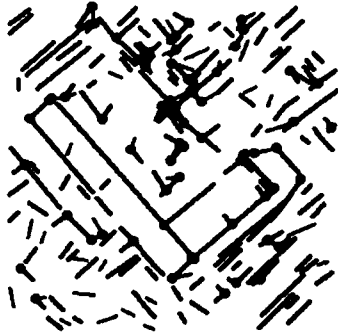


Figure 4. Linear structures and junctions

Parallels and U-structures

Structures in urban scenes like buildings, roads and parking lots are often organized in regular grid-like patterns. These structures are composed of parallel sides. As a consequence, for each significant line-structure detected in the scene, there is not one but many lines parallel to it. For each line, we find lines that are parallel and satisfy a number of reasonable constraints. Note that the formation of a *parallel structure* also aids in the formation of new *lines*, as they suggest extension and contraction of the parallels to achieve full overlap.

When the two lines in a parallel structure have their ends aligned, they strongly suggest the presence of a line with which the parallel structure would form a *U-structure*. Even if the third line does not exist in the set of *lines*, we hypothesize it and generate the U-structure.

Rectangles

Rectangle structures are generated from the U-structures. The rectangles formed in our example are shown in Figure 5. In practical applications this number can be reduced by restricting the formation of rectangles on the basis of size, as a function of image resolution, for example. Rectangles are also generated from matching junctions along the direction of illumination (see strong junctions in section 5.) We hypothesize the missing portions of a rectangle having a corner with a matching shadow corner.

4 Selection of Hypotheses

After the formation of all reasonable rectangles, a selection process is applied to choose rectangles having strong evidence of support and having minimum conflict among them. Our previous system used a Constraint Satisfaction Network (CSN) [Mohan and Nevatia, 1989]. Here, we use a different method which seems

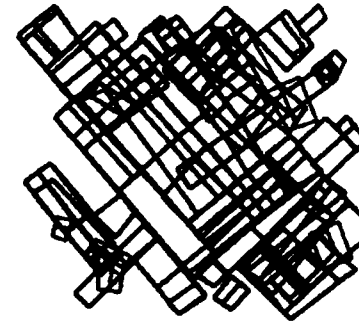


Figure 5. Rectangle hypotheses generated

to give much more predictable results.

Our new system uses two kinds of criteria: *local selection criteria* and *global selection criteria*. Local selection criteria determine whether or not a rectangle is "good" based on the local supporting evidence. Only good rectangles are retained for global selection. It is possible that some of the good rectangles retained after the local selection are mutually contained or duplicated or overlapped with some other good rectangles. Global selection criteria select the best consistent rectangles from good rectangles.

We apply local selection criteria and global selection criteria differently. Local selection criteria (evaluation criteria) work together to evaluate the goodness of a rectangle, while global selection criteria work separately. Each global selection criterion acts like a *filter*. The set of retained rectangles pass through all filters and the set of rectangles coming out from the last filter will be the set of rectangles selected by the selection process.

The local selection criteria are used to remove rectangles formed using weak evidence. For each rectangle the evaluation criteria compute a goodness value. If this value exceeds a given threshold, the rectangle is selected, otherwise the rectangle is removed.

Every evaluation criterion is weighted according to its importance. The goodness of a rectangle is then measured by the sum of the weighted values returned by the evaluation criteria. The problem of measuring the goodness of a rectangle now becomes a problem of finding and formulating good evaluation criteria, and how to assign appropriate weights.

Whether a rectangle is good or not depends on evidence of support. We distinguish between *positive evidence* and *negative evidence* of support for a rectangle. The positive evidence we use includes the presence of edges, corners, parallels, and shadows. The negative evidence includes the presence of lines crossing any side of a rectangle, existence of L-junctions or T-junctions in any side of a rectangle, existence of overlapping gap on opposite sides of a rectangle, and displacement between four sides of a rectangle and its corresponding edges support. Negative evidence is as important as positive evidence, because they help us to remove those rectangles which are less likely to be part of buildings.

Good rectangles surviving local selection may compete with each other. For example, some rectangles

could share the same edge or corners support and some rectangles might overlap with each other. The goal of global selection criteria is to select a minimum set of rectangles which best describe the rectangular composition of the scene.

Global selection criteria examine overlapping rectangles and choose one if appropriate. The selection is based on relative properties of each rectangle, the amount and kind of overlap, and whether they share support or not. Note that a rectangle fully contained in another is not necessarily removed. If a rectangle does not overlap with any other rectangles then it is not in competition, and it remains. If available, some of the shadow evidence is used in this process.

The rectangles selected in our Ft. Hood example after both the local and global selection criteria have been applied are shown in Figure 6.

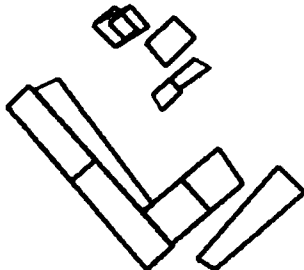


Figure 6. Selected Rectangles

5 Verification of Hypotheses

The purpose of verification is to validate the selected hypotheses to correspond to buildings. Our validation step segments the objects, generates a description of the shape of the structures and derives a 3-D model.

5.1 Shadow Analysis

By shadow analysis we mean the establishment of correspondences between shadow casting elements and shadows cast, and the use of these correspondences to verify and model 3-D structures. We assume that the sun angles are given and that the ground surface in the immediate neighborhood of the structure is fairly flat and level. The shadow casting elements are given by the sides and junctions of the selected rectangle hypotheses. The shadow boundaries are located among the lines and junctions computed earlier from the image.

There are a number of difficulties that prevent the accurate establishment of correspondences however. Building sides are usually surrounded by a variety of objects such as loading ramps and docks, grass areas and sidewalks, trees, plants and shrubs, vehicles, light and dark areas of various materials. Nearby structures may reflect light into the shadowed areas making the objects in it more visible, and so on. To deal with these problems we have adopted the following definitions, criteria and geometric constraints to analyze the shadows adjacent to rectangles (see Figure 7):

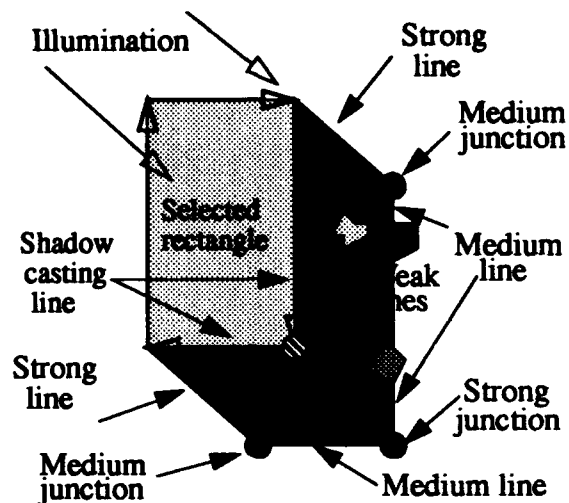


Figure 7. Shadow features

Strong Junctions

Matching junctions along the direction of illumination, having a consistent *shape* and a consistent *attitude*. These junctions constitute the strongest monocular cue to the presence of a 3-D structure. We use them also to form and select rectangle hypotheses.

Strong Lines

Vertical building edges cast shadow lines in a direction similar to the direction of the projection of the sun rays. We use this evidence also during hypotheses selection.

Medium Lines

The rectangle sides that are supposed to cast shadows must have corresponding shadow lines.

Medium Junctions

The junctions formed by strong and medium lines, found along the direction of the strong lines.

Weak Junctions and Lines

Junctions and breaks in the shadow boundaries between the strong and weak junctions.

Strong Regions

Dark regions surrounded by strong and medium junctions. We require that this region be darker than the rectangle region regardless of their gray level.

Weak Regions

In the absence of geometric correspondences of junctions and lines, a dark region adjacent to rectangle, consistent with the direction of illumination.

5.2 Shadow Process

The shadow process consists of four steps:

Extraction of Potential Shadow Evidence

Potential shadow evidence consists of lines, junctions

and intensity statistics. We extract the following:

- Lines parallel to the projection of the sun rays. They represent potential shadow lines cast by vertical edges of 3-D structures in the image.
- Lines having their dark side on the side of the illumination source are potential shadow lines.
- Junctions among the lines above.
- Pixel statistics to compare relative brightness.

The potential shadow lines and junctions extracted from the lines in our Ft.Hood example are shown solid in Figure 8. The underlying edges are shown in gray.

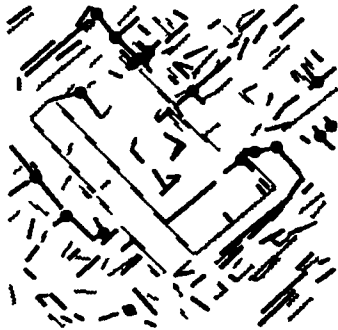


Figure 8. Potential shadow lines and junctions

Search for Shadow Evidence

For each rectangle we look in a search window (dashed lines in Figure 9) and collect all the potential shadow evidence in it. The search distance is arbitrarily chosen as a function of the maximum expected building height and the sun incidence angle. There is the possibility that lines, not relevant to the current rectangle, be included. They however, have a reduced effect in the presence of the real evidence.

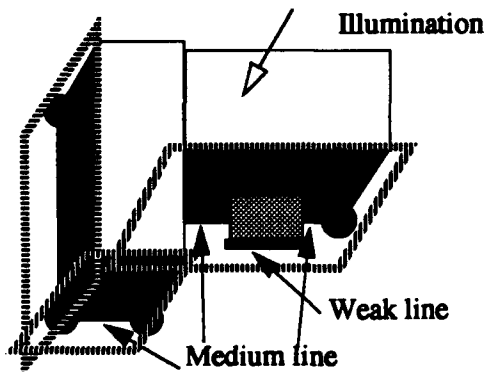


Figure 9. Windows to search for shadows

We favor medium and weak lines that are parallel to the rectangle side. In some cases there may be various sets of lines, all parallel to the building side but at various distances from the rectangle side. This is actually a common occurrence since many side walks, grass areas, streets, vehicles and so on, will be found to be arranged or located parallel to building sides. In this case we choose those shadow lines at the distance from the rectangle side such that the sum of their lengths is greater, but not exceeding the length of the rectangle. We deter-

mine the width of the shadow by averaging the distance to the lines selected. The selected evidence is then considered to surround the shadow region. We compute the mean intensity of this region and compare it to the rectangle region. The evidence collected for both sides is combined to give the evidence for the rectangle.

Evaluation of Shadow Evidence

We evaluate the shadow evidence and give a confidence value as a weighted sum of the evidence of strong junctions, medium junctions, strong line, weak lines, strong and weak regions. We designated five levels of confidence. Each level of confidence requires that a minimum amount of the different kinds of evidence be present. Very high confidence requires that every kind of evidence be detected. Very low evidence is reported when no geometric correspondences can be established but the presence of a region, adjacent to and darker than, the rectangle region itself, is found.

The rectangles selected on the basis of shadow evidence are shown in Figure 10.

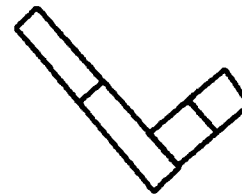


Figure 10. Rectangles with evidence of shadows

Use of Shadow Evidence

The rectangles validated by shadows are used to give the footprint of buildings or portions of buildings. The shadow widths are used to estimate their height. The result is an elevation map encoding the height computed for each rectangle. A 3-D rendered view computed from the rectangles verified in our Ft. Hood example is shown in Figure 11.



Figure 11. 3-D view from another viewpoint

6 Results

We have tested our technique on many images from the Ft. Hood site and from a modelboard site. We selected a few to demonstrate the performance of our system. In the remaining figures (except Figure 16), (a) is the image, (b) the line segments, (c) the lines and junctions, (d)

the rectangles, (e) the selected hypotheses, and (f) the hypotheses verified by shadows. In particular, note figure (e), the excellent performance of the new selection technique. In the absence of shadow information, the selected rectangles can be matched by our system if stereo views are available, thus providing verification and a 3-D model.

Figure 12 shows a set of four buildings and part of another. The difficulty here is with the building with the patterned arrangement of small objects on the roof. The shadows cast by these reach one side of the building causing it to be fragmented. The shadow occluding the top left corner of the building and the poor boundary definition on the top right are also a source of difficulty. The strong shadow cues however help form rectangle hypotheses for most of the building

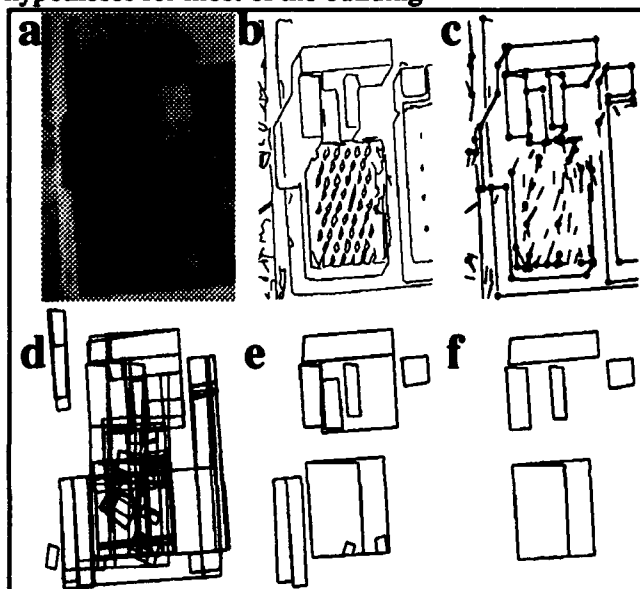


Figure 12. Modelboard - Scene 1

In Figure 13 the small building on the top left corner of the large one is detected separately. Note that portions of buildings not in full view are also detected.

Figure 14 shows two dark buildings. The boundaries between buildings and shadows in cases like this has low contrast and are difficult to detect.

Figure 15 shows a complex building with numerous rectangular components on the roof. We are able to exploit the presence of strong shadow evidence. It allows the system to form a hypothesis for the entire building in spite of the broken and fragmented boundaries. Note that the selection mechanism is able to select most of the rectangular components on the roof as well. In this example the shadows are well defined, and it is possible to measure their width accurately. Figure 16 shows a 3-D rendered view of the building.

Figure 17 shows the same building from an oblique view (30°). The roof remains a rectangle with small perspective effects that the system is able to tolerate. Processing oblique views is the subject of our current work.

Figure 18 shows a building in Ft. Hood, where some

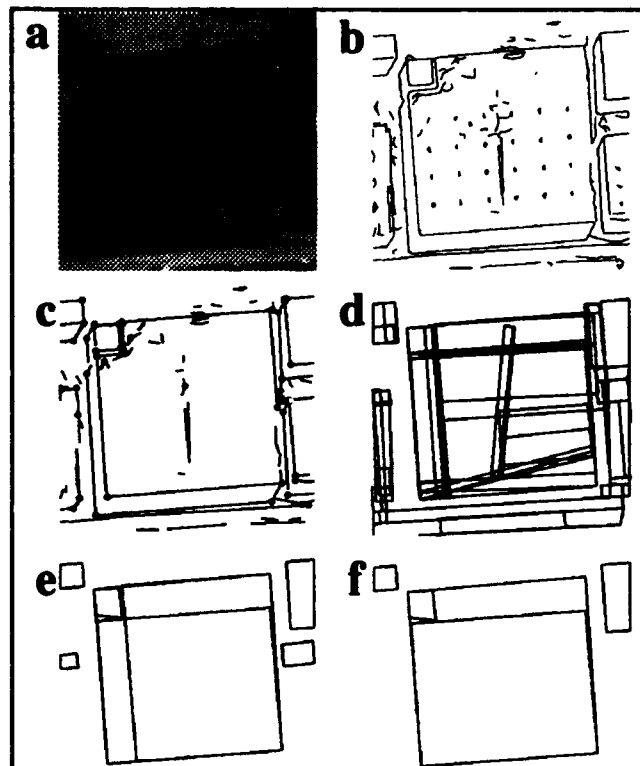


Figure 13. Modelboard - Scene 2

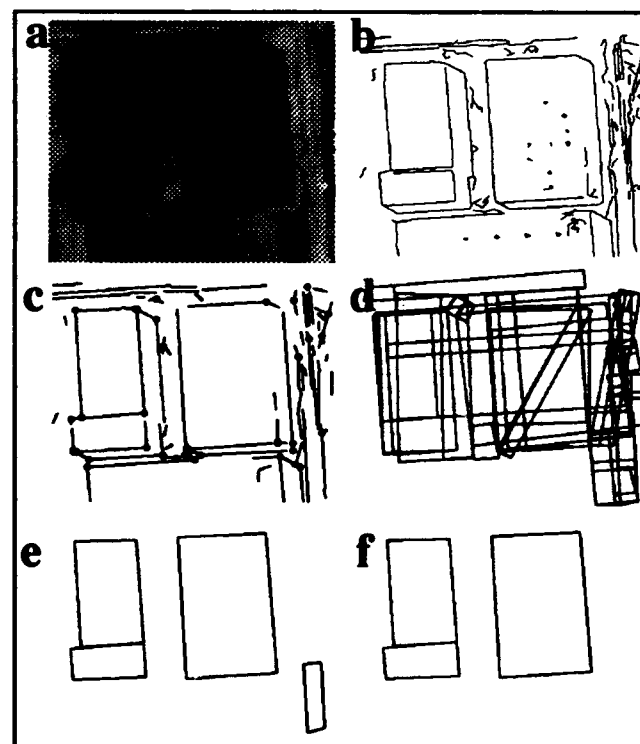


Figure 14. Modelboard - Scene 3

of the details of one of its sides is visible, apparently doors. These and the vehicles parked on the other side result in highly fragmented boundaries. The rectangles verified by shadows include one that is formed from various aligned parked trailers which collectively cast a shadow. The small rectangle on the bottom has a strong

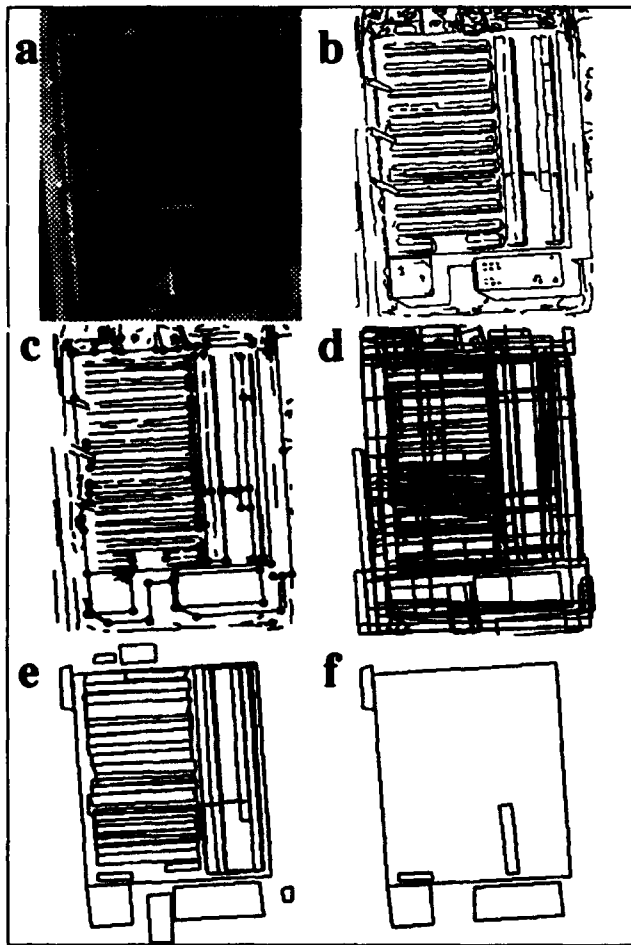


Figure 15. Modelboard - Scene 4

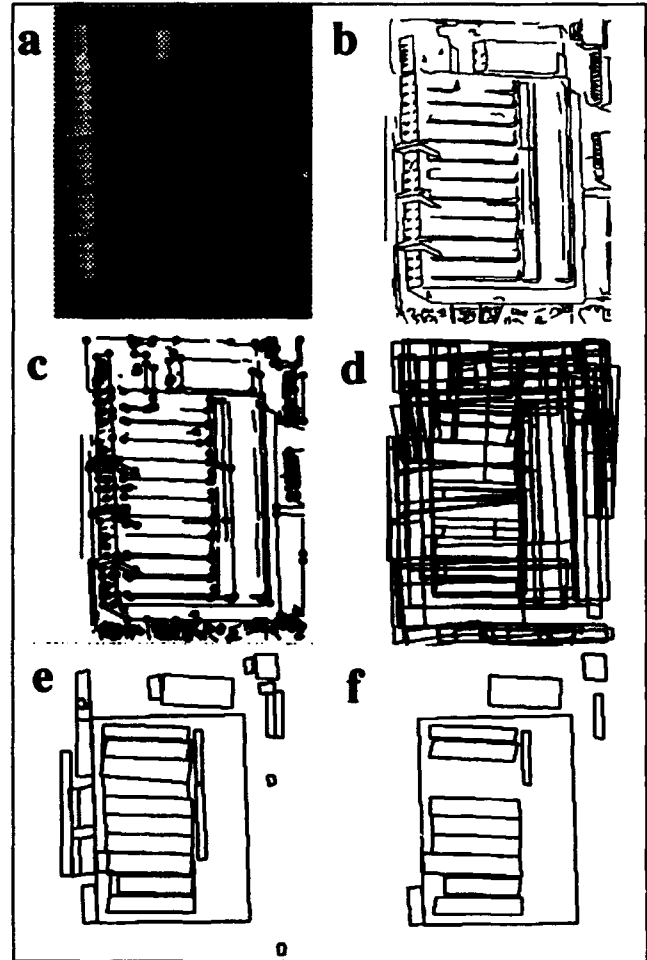


Figure 17. Modelboard - Scene 4 (oblique)

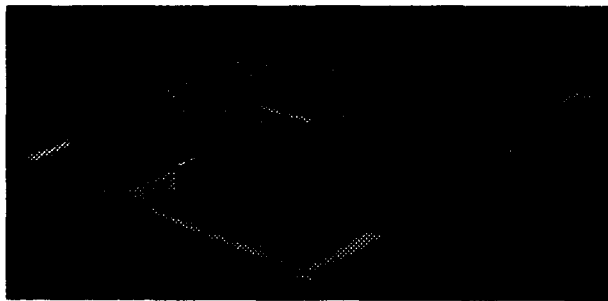


Figure 16. 3-D view from another viewpoint

shadow junction corresponding to an actual narrow shadow cast by a vehicle. The lower wing of the building has a strong line and a corresponding medium junction. The rest of the shadow is diffused and only gives a "dark" region adjacent to the building wings.

In Figure 19 the I-shaped building has no strong evidence of shadows. The rectangles are weakly validated on the basis of a strong region which up a given maximum search distance remains "strongly" dark.

In Figure 20 shows a not uncommon situation. The building is surrounded by rectangles representing grassy areas and sidewalks parallel to the building. The only shadow evidence is the dark region adjacent to the

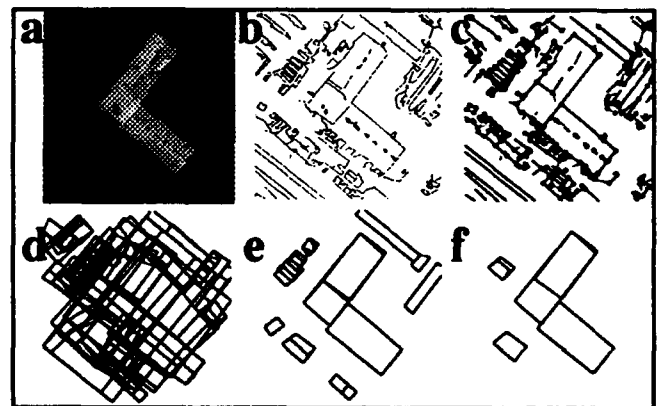


Figure 18. Fort Hood - Scene 2

building. Note that the other rectangles have adjacent bright regions, namely, the building itself and its wings.

Figure 21 shows a group of small buildings arranged in a parallel fashion, and surrounded by other parallel structures. In spite the large number of hypotheses the system is able to select the relevant ones.

7 Current Work: Oblique Views

In the scenes analyzed, many of the objects have restricted shapes and often the viewpoint is restricted. For some applications, it is necessary to integrate informa-

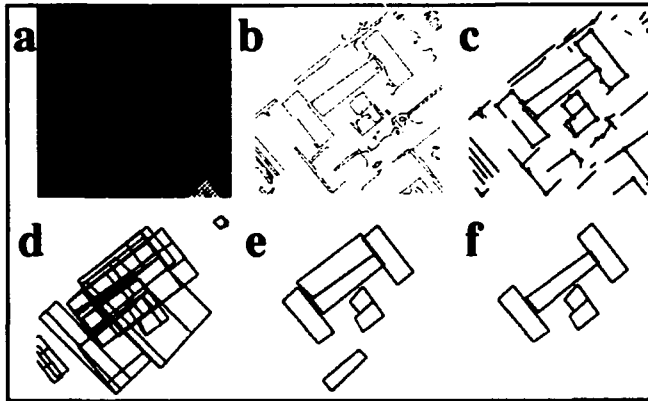


Figure 19. Fort Hood - Scene 3

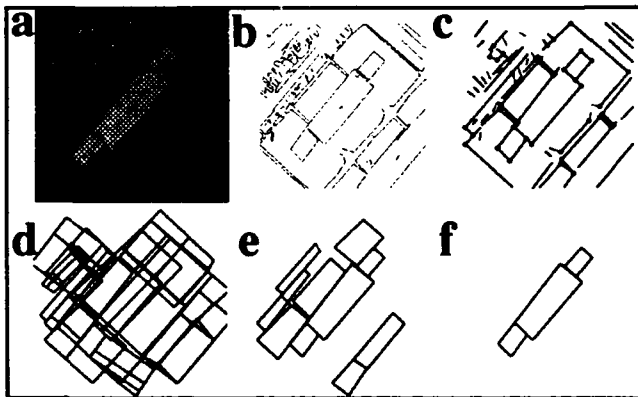


Figure 20. Fort Hood - Scene 4

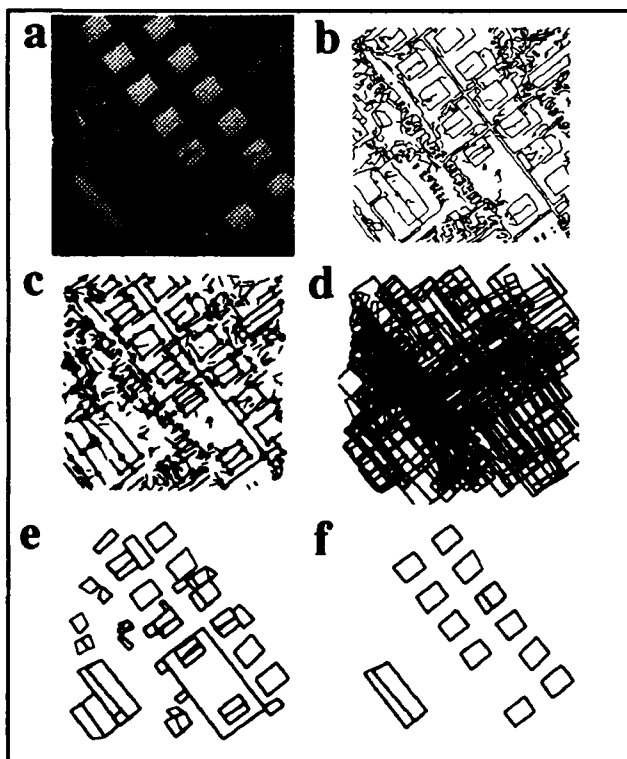


Figure 21. Fort Hood - Scene 5

tion extracted from images of a scene acquired from various viewpoints or acquired through various types of sensors. We are currently performing extensive testing of our system and reviewing our methods to determine the feasibility of relaxing viewpoint restrictions. We have begun investigating orthogonal trihedral vertices (OTVs) in oblique views. If we continue to assume that we restrict the shape of the objects to rectangles, the most significant change is that right angles in the real world no longer necessarily project onto right angles in the image.

References

- [Canny, 1986]. J. Canny, "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679-698, November, 1986.
- [Chung and Nevatia, 1992] C. Chung and R. Nevatia, "Recovering Building Structures from Stereo," *In Proceedings of the IEEE Workshop on Applications of Computer Vision*, pages 64-73, Palm Springs, Calif., December 1992.
- [Huertas, 1983]. A. Huertas, "Using Shadows in the Interpretation of Aerial Images," University of Southern California USC-ISG Technical Report 104, October 1983.
- [Huertas and Nevatia, 1988]. A. Huertas and R. Nevatia, "Detecting Buildings in Aerial Images," *Computer Vision, Graphics and Image Processing*, 41(2):131-152, February 1988.
- [Irvin and McKeown, 1989]. R. Irving and D. McKeown, "Methods for exploiting the Relationship Between Buildings and their Shadows in Aerial Imagery," *IEEE Transactions on Systems, Man and Cybernetics*, SMC 19(6): 1564-1575, November/December 1989.
- [Liow and Pavlidis, 1990]. Y. Liow and T. Pavlidis, "Use of Shadows for Extracting Buildings in Aerial Images," *Computer Vision, Graphics and Image Processing*, 49: 242-277
- [Mohan and Nevatia, 1989]. R. Mohan and R. Nevatia, "Using Perceptual Organization to Extract 3-D Structures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1121-1139, November 1989.
- [Nevatia and Babu, 1980]. R. Nevatia and R. Babu, "Linear Feature Extraction and Description," *Computer Vision, Graphics and Image Processing*, 13:257-269, 1980.
- [Venkateswar and Chellappa, 1990]. V. Venkateswar and R. Chellappa, "A Framework for Interpretation of Aerial Images," *In Proceedings of the International Conference on Pattern Recognition*, pages 204-206, Atlantic City, NJ, June 1990.

Section IV

Unmanned Ground Vehicle (UGV)

The JISCT Stereo Evaluation*

Robert C. Bolles, H. Harlyn Baker, and Marsha Jo Hannah

Artificial Intelligence Center, SRI International
333 Ravenswood Ave., Menlo Park, CA 94025
(bolles@ai.sri.com baker@ai.sri.com hannah@ai.sri.com)

Abstract

The results of the "JISCT" Stereo Evaluation (named after the five groups contributing imagery: JPL, INRIA (in France), SRI, CMU, and Teleos) are presented. The goals of this evaluation, which was the first phase of a multiphase evaluation process, were (1) to get an initial estimate of the effectiveness of current stereo techniques applied to Unmanned Ground Vehicle (UGV) tasks, (2) to identify key problems for future research, and (3) to debug the evaluation process so that it can be repeated with a larger group of participants. SRI collected 49 pairs of images, distributed them to the five participants, and received complete results from three groups — INRIA, SRI, and Teleos. SRI compared the results by interactively analyzing them and automatically gathering statistics.

We were surprised by the completeness of everyone's results. On the eight image pairs that we thought were the most representative of UGV tasks, the techniques computed disparities for as much as 87% of the points with only a few "spike" errors and some scattered regions of points without matches. Although the missing points (and mistakes in the reported matches) could cause problems for vehicle navigation, this level of completeness is an indication that there is a solid basis for building a passive ranging system for an outdoor vehicle. On the other hand, none of these techniques have "solved the stereo problem" — we selected a number of important areas for future research, including filtering out gross errors and handling the wide dynamic range of intensities common in outdoor imagery.

1 Introduction

Stereo analysis, which for a long time had been viewed as an interesting, but too-costly-to-be-practical technique, has emerged as a viable tool for realtime applications such as vehicle navigation. This has happened

for two reasons. First, advances in hardware have made it practical to compute stereo matches "in real time." And second, advances in algorithm development have made it possible to correctly match large portions of outdoor scenes.

An important next step in the development and use of practical stereo systems is the characterization of their capabilities. Potential users, such as system integrators and automatic task planners, need to know their computational requirements, their speeds, their precision, their mistakes, and so forth, in order to model their behavior and reason about their use. With this in mind, SRI, JPL, and Teleos began a multiphase evaluation process last year within the Unmanned Ground Vehicle (UGV) Project. The first phase of that evaluation has been completed, and the second phase has begun. This paper describes the results of the first phase.

The overall plan for our complete evaluation process is to pursue a three-pronged approach, including analytic models, qualitative "behavioral" models, and statistical performance models. The analytic models would be used to estimate such things as the expected depth precision computable with a specific camera configuration. The qualitative models would be used to identify key problems for future research, for example, detection of holes, analysis of shadowed regions, and measurement of bland areas. The statistical models would be used to produce quantitative estimates of such key factors as the smallest obstacle detectable at a specified distance. SRI has taken the lead in the qualitative evaluation; JPL has taken the lead in the quantitative analysis.

For the qualitative analysis, we decided to start by examining a small number of techniques in order to debug the process, and then expand the evaluation to include a much larger set of participants. The goals of the first phase were to get an initial estimate of the effectiveness of current stereo techniques applied to UGV tasks and, from this, to identify key problems for future research.

One of the high-level guidelines we adopted was to develop and maintain an atmosphere of cooperation and constructive criticism among the researchers participating in the evaluation. Without this we would not be

*Supported by Advanced Research Projects Agency Contract DACA76-92-C-0003.

able to focus on our ultimate goal of producing a sequence of increasingly capable stereo systems. To help establish a cooperative atmosphere, we decided to concentrate on the positive aspects of each algorithm and highlight ways to strengthen existing techniques, realizing that they were developed for different domains and different applications. We also decided to share all the raw results with the participants so they could duplicate our analysis or develop their own.

For the first phase of the qualitative evaluation, SRI collected imagery from five groups, JPL, INRIA (in France), SRI, CMU, and Teleos (hence the name "JISCT" for the first evaluation phase); selected 49 pairs for analysis; converted them into a standard format; distributed the dataset to the five groups for processing, along with an extensive set of instructions; collected the results; characterized them; and finally distributed the results and the associated report to the participants.

We intentionally asked each group to process a large number of pairs (10 training pairs and 45 "test" pairs ... 6 pairs were in both the training and test sets; we made an administrative mistake on one of the test pairs, reducing the total to 44), because we wanted to force them to establish a standard algorithm that was automatically applied. As a result of this, there are now four groups around the world that can readily apply end-to-end stereo techniques to new data and compare their results. As part of the second phase we hope to expand this community to 10 or more groups. This process is opening up a new form of interaction within the computer vision community that we feel will help stimulate advances and reduce redundant development.

In the instructions to the participants, we asked each group to produce several results for each matched point in addition to its computed disparity. For each point we asked for an x and a y disparity, an estimate of the precision associated with each reported disparity, an estimate of the confidence associated with each match, and an annotation for each unmatched point, indicating why the technique could not find a match. Possible explanations for no match included "area too bland," "multiple choices," and "inconsistent with neighbors." Although none of the groups produced all this additional information (they all produced some of it), we felt that it was important to begin the process with the goal of producing this auxiliary information, which will be invaluable for the higher-level routines using the stereo results. We foresee a time in the not too distant future when the calling routine will use the precisions, confidences, and annotations to actively control the sensor parameters for the next data acquisition step. For example, if the current stereo results contain a large region of points without disparities and the image region is quite dark, the controlling routine could open the irises or increase the integration time to reexamine these dark regions.

Four groups returned results and write-ups to SRI --

Teleos, SRI, and two from INRIA. One of the INRIA sets was from a technique that locates linear features and then matches these features. Since this technique reports only disparities along the matched edges, it was not possible to directly compare its results to the others. Therefore, we concentrated our analysis on the three correlation-based algorithms.

Each participating group analyzed its own results. In addition, Harlyn Baker and Marsha Jo Hannah of SRI analyzed the results from all the groups on all 44 pairs and wrote short reviews of them. In the full report [Bolles, Baker, & Hannah], their comments are included as appendices. These comments, plus the automatically compiled statistics, form the core of this evaluation.

Initially, we were a little reluctant to compute and publish statistics that may be taken out of context. On the other hand, statistics, if reported with sufficient caveats, can provide a convenient basis for comparing techniques. In this paper, we summarize the qualitative results and quantitative statistics. The validities of both are limited by the dataset, which implicitly defines the range of data for which the conclusions directly apply, and by the analyzers, who naturally focused on issues they were most interested in.

This paper is organized as follows. In Section 2, we briefly describe the key strategies and parameters of the three principal techniques, highlighting their similarities and differences. In Section 3, we describe our experimental procedure. In Section 4, we present the automatically gathered statistics, which we refer to as the believe-everything-they-tell-you statistics because they are based on the number of "reported" disparities in specified regions of the test data, not on the number of "correct" disparities. In Section 5, we summarize our qualitative analysis and briefly discuss open issues for future research. In Section 6, we conclude with an evaluation of the JISCT evaluation and make some suggestions for the next step in the evaluation process.

2 Technique Summaries

We evaluated three techniques, whose key aspects are highlighted below.

2.1 INRIA

This technique was originally implemented as part of a European space project to produce three-dimensional models of scenes containing rocks and sand. It is implemented in C on a Sun. A similar technique is implemented on a Connection Machine (by Pascal Fua) at SRI. Key aspects are

- The algorithm computes a disparity for every pixel in an image by matching patches (usually 11×11 pixels) at one or two image resolutions, independently.

The basic algorithm "INRIA-1" matches only at one resolution.

- The technique uses an approximation to normalized correlation, referred to as C5, because it can be implemented efficiently using a sliding computation of the basic sums.
- The algorithm searches only along epipolar lines, which are assumed to be horizontal.
- The algorithm expects a range of disparities to be specified for each image pair to be analyzed.
- The technique verifies all matches by independently matching patches from the left image in the right image and patches from the right image in the left image. If the match for a patch from the left image is not mapped back to within a pixel of its location in the left image, the point is not assigned a disparity.
- The technique computes a subpixel location for each match by fitting a second-order curve to the correlation values surrounding the best match.
- After computing disparities for as many pixels in the left image as possible, the algorithm filters out isolated matches by morphologically shrinking the regions of matches. It typically shrinks the regions three times, grows the result three times, and then ANDs this result with the original image of results. This process can erase regions as large as 6x6 pixels.
- The algorithm computes a confidence value for each disparity by differencing the heights of the two highest matching peaks.
- The technique estimates the precision of a disparity value by fitting a Gaussian to the matching peak, using its standard deviation as the precision measure.
- The technique does not attempt matches near the edges of an image.
- The second set of results provided for this evaluation often was produced by matching at two image resolutions and picking the highest resolution for which there was a valid match.

2.2 SRI

This stereo system has evolved over 20 years, beginning with early Martian Rover research, migrating into the aerial mapping domain, and now coming back to ground-level analysis. Its goal has been to produce a set of high-quality matches from a wide range of (possibly uncalibrated) imagery. The algorithm is a multi-stage process that uses one matching technique to get a few solid matches at high-information points, and then

uses these matches to guide another matching technique, whose results become anchors for yet another technique, etc, with culling of mistakes occurring at many levels. At each stage, the algorithm acquires more supporting matches to suggest limits for the disparity search, so the algorithm can attempt to match points that have less "interesting" information, using less hierarchy. For this evaluation, code was added to produce "dense" matches; this included stages that grow regions of matches around previously matched points, and fill in a regular grid of matches. In total, the standard algorithm for this evaluation involved seven stages of matching and three filtering steps. The algorithm is implemented in C on a Sun; speed has not been a priority.

Some key aspects are

- The algorithm applies a version of hierarchical matching for each point that it analyzes. At the early stages of the process, it uses all available image resolutions, starting at the coarsest, using the match found at that level to predict the location of the match at the next finer level, then refining it, and so forth. At the final stage, where the dense grid of points is computed, the algorithm uses only one or two levels.
- At each image resolution (level), the algorithm does a two-dimensional search near the epipolar line and then hill-climbs around the best match. The epipolar lines can be at any angle in the second image, and if there is no camera model (due to bad matches at early stages, or because the camera isn't modelable by a pinhole camera), the algorithms search over areas—(dx,dy) boxes—defined by surrounding matches.
- The algorithm uses normalized cross correlation (correcting for a linear intensity change from image to image) on 11x11 patches typically. Later stages, such as the region-growing step, can use smaller patches. The final match includes a subpixel estimate of the disparity, computed by fitting two parabolas to the nearby correlation values.
- Each match from one image to another is verified by applying the same technique to match back into the original image. If the return match is not within a pixel of the original point, the match is discarded as unreliable.
- The algorithm applies several other "filters" to weed out mistakes, including a threshold on interest value, thresholds on relative and absolute correlation values, tests for matches outside an image, and tests for unusual disparity values within a region of the image.
- Later stages of the algorithm use previously computed disparities in the neighborhood of a new point

to be matched, to specify the range of disparities to be considered. The neighborhoods are typically large, beginning at 1/4th of the image area, and gradually reducing to 1/64th of the image for this experiment. This technique assumes that the scene is composed of relatively large continuous surfaces.

- Since a confidence for each match was requested for this experiment, one was supplied by computing the ratio of the correlation value to the autocorrelation threshold.

2.3 TELEOS

This technique has been designed for efficient implementation and recently has been geared toward active vision in which the basic stereo process matches 100 to 200 selected points in a 1/30th of a second. It is implemented on a combination of two special boards and a Datacube system. For this evaluation, however, the hardware was not available and so a Lisp version of the algorithm (running on a Lisp Machine) was used. Some key aspects are

- The algorithm uses large correlation windows (ranging from 24x24 to 96x96 pixels).
- The algorithm computes binary correlation values from the Laplacian of Gaussian of the original images.
- The algorithm analyzes the data only at one resolution. It automatically selects the size of the convolution operators by analyzing the peak shapes of matches at 25 points in each new image pair. It selects the smallest window size that produces a significant difference between the heights of the top two highest peaks.
- At each point in the image, the algorithm starts with the disparity computed for the neighboring pixel and tries to locate a match at a similar disparity. A serpentine search, which analyzes the first row from left to right, the second row from right to left, and so forth, is used in order to reduce the computation time on the Lisp Machine.
- The algorithm searches off the epipolar line for the best match.
- The algorithm also examines the effect of skewing the patch being matched. It analyzes skews ranging from -.5 pixels per line to +.5 pixels per line. This analysis is applied only at the end of the search when the best match has been selected.
- The algorithm estimates a subpixel disparity value by fitting a quadratic function to the best peak.
- The algorithm does not try to match points near the edges of an image.

3 Experimental Procedure

The goal of this initial evaluation was to produce a qualitative characterization of the capabilities of current stereo techniques applied to UGV tasks. The intent, as stated in the instructions distributed to each participant, was to produce a description such as the following:

On the 44 image pairs in the database our techniques correctly measured disparities to 65% of the points on the ground and 40% of the points on obstacles, such as trees, bushes, and rocks. The top five problems for our techniques were dynamic range, holes, bland areas, repeated structure, and poor range resolution. We estimate that these problems occur in the UGV scenarios with frequencies of ...

The idea was to produce a characterization that would focus future work on key UGV problems.

Our basic approach to developing this type of characterization was to apply the techniques to a large dataset, visually display the results in ways to highlight unusual events, gather basic statistics, and where possible, summarize our observations in descriptions that link observed behaviors to aspects of the techniques.

To start the process, SRI compiled a database of 49 image pairs from JPL, INRIA, SRI, CMU, and Teleos. We converted the images into a standard format and then distributed them to the five contributing groups for analysis. The groups were instructed to use 10 pairs as a training set, "freeze" their algorithm, and then process the whole set of 45 pairs. Results and commentary from four stereo systems were returned to SRI — Teleos, SRI, and two from INRIA. One of the INRIA sets, using edge-based feature analysis, could not easily be compared with the others. We concentrated our analysis on the three correlation-based system results.

To assist in the analysis of the results, SRI developed two sets of routines, one to gather statistics and one to display the disparities in a variety of ways. Since we did not have ground truth for the distributed imagery, we were not able to compare the computed disparities with objective values. However, we were able to gather statistics on two of the three types of mistakes that we are interested in by outlining special regions in the imagery and counting the occurrence of results within these regions.

We made a distinction between the following three types of mistakes:

False Negatives: No disparities computed for points that should have results.

False Positives in Unmatchable Regions: Disparities reported for points that don't have matches in the second image, for example, points occluded in one

image or points out of the field of view of one of the images.

False Positives in Matchable Regions: Incorrect disparities reported for matchable points.

By interactively outlining regions of occluded points, regions of points out of the field of view of the second image, and regions of points in the sky, we were able to directly measure statistics for the first two types of mistakes. In addition, we outlined regions corresponding to expected problems, such as dark shadows, foliage, and bland areas. In this way we could gather statistics on the behavior of the algorithms on these special problems.

As part of the initial instructions we asked each group to extend its algorithm to produce an image of annotations that summarizes the result of the analysis, pixel by pixel. At each pixel we asked for a code from the following list:

- 0: no match attempted
- 1: matched fine

NO MATCH BECAUSE

- 2: too bland, no information to key on
- 3: low match value (e.g., correlation value)
- 4: multiple choices (ie, repeated structure)
- 5: back-match inconsistency
- 6: point out of camera's field of view
- 7: point occluded by an object in the scene
- 8: point too far off the epipolar line
- 9: point inconsistent with neighbors
- 10: other

The reason for requesting these codes is to encourage future algorithms to provide this additional information, which can be used by the higher-level vision techniques to decide what should be done next. For example, if no results are reported for a region directly ahead of the vehicle and the region is too bland and very dark, one option might be to open the irises on the cameras (or increase the integration time) in order to see into the dark area.

INRIA reported codes of 1 and 10; SRI reported all codes except for 4 and 7; and Teleos reported codes of 1, 2, and 3. Therefore, we were able to count the number of matches attempted in each region and the number of disparities reported.

To estimate the frequency of incorrectly reported disparities (the third type of mistake), we either compared them to interactively selected values or located an aberration in the local pattern of disparities when they were displayed on the screen. We experimented with a variety of display techniques, including displaying the disparities as color-coded dots in stereo, heights above a three-dimensional "ground" plane, and disparity-displaced vertical lines. We are continuing to look for better ways

to display three-dimensional results, because most current techniques encourage the human eye to "smooth over" differences, making the results look better than they actually are.

4 Statistics Summary

The statistics that we refer to as believe-everything-they-tell-you statistics are based on the number of reported disparities in specified regions of the test data. These statistics do *not* distinguish between "correct" and "incorrect" disparity values, just reported values and unreported values. They do, however, provide enough information to estimate three important quantities, the number of false negatives (matchable points that were not assigned a disparity), the number of false positives occurring in unmatchable regions, and the number of matchable pixels that were assigned disparities.

To help focus attention of key areas of the test data, we interactively outlined regions in the left images of 20 of the 44 image pairs (see Figure 1 and Figure 2). One of the most important regions is what we called "matchable-data." It eliminates several types of points that do not have matches in the right image, including null bands that do not contain grayscale data (but are included in the images to fill them out to a standard size, such as 512 by 512 pixels) and pixels that are out of the field of view of the right camera. In the 20 images we examined, the percentage of unmatchable points ranged from 4.3% to 46.0% and averaged 12.3%.

The statistics were gathered by a program that counted the number of disparities (dx disparities) reported in the specified region (or the whole image, if that was appropriate).

Figure 3 shows the results on all 44 image pairs. Note that

- The dataset contains a wide variety of imagery; some of it is realistic (containing dirt roads and cross-country scenes) and some is designed to test the algorithms along one dimension, such as baseline and noise tolerance. Some of the imagery is even trick imagery (the shoe images from CMU).
- The numbers in parentheses after each group's name (along the top of the table) indicate the number of test pairs in the dataset from that group.
- The INRIA-2 results are in parentheses because different parameter settings were used for different image pairs. However, the usual change was for the technique to match at two spatial resolutions instead of just one, and then combine the results. If a second set of parameters was not tried for a pair, we left the entry blank and used the INRIA-1 results in our computation of INRIA-2's average.

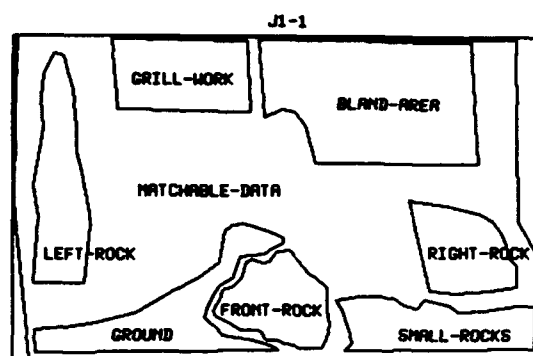
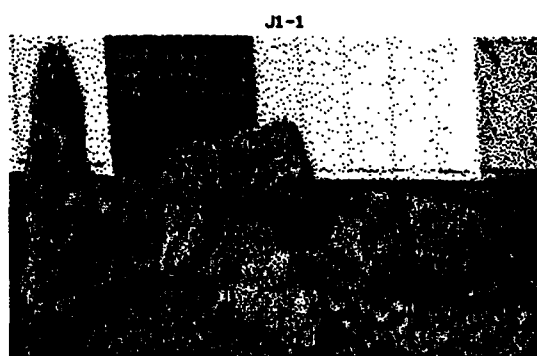


Figure 1: Interactively outlined, special-interest regions for the J1 image pair from INRIA.

- If we did not outline a “matchable-data” region for a pair, we used the full-image statistics in our computations. This reduces the effectiveness totals somewhat (possibly by as much as 7%).

Given the diversity of the data, we were pleased with the completeness of the results.

In order to examine the behavior of the techniques on typical UGV imagery, we selected the eight images from the dataset that were the most appropriate for UGV tasks and collected statistics on that subset. Figure 4 shows the results on these data. The INRIA-2, SRI-2, and Teleos-1 techniques performed well, computing disparities for 86 or 87% of the matchable points. Note, however, that these images did not contain difficult obstacles, such as holes, ditches, and small rocks—the obstacles were large rocks, bushes, and trees.

Figure 5 shows the results on the 17 large obstacles in the dataset. The techniques did an excellent job of detecting these objects, which stick up above the ground—they only had a little trouble in shadowed regions on them.

With respect to shadows, the techniques had a significantly harder time computing disparities for points in shadowed regions than in sun-lit regions. Figure 6 shows the results for points in shadows.

The techniques also had trouble with bland regions, as expected. Figure 7 shows the results on these areas. The techniques typically computed results around the edges of the regions—the larger the correlation windows, the more points were computed, because correlation windows naturally extend matches into the interior of bland regions by about half their diameter.

There are several potentially important problem areas that were not covered in this initial dataset, including holes, sand, small- to medium-sized rocks and bushes, reflective surfaces (water or windows), and moving objects. One of our goals for the second phase of this evaluation is to include examples of these problems.

5 Qualitative Analysis

We were surprised by the completeness of everyone's results. Even though the dataset contained a wide range of imagery, including some sequences designed to stretch the analysis along specific dimensions, such as noise tolerance and disparity range, the techniques computed disparities for 64% of the matchable points. On the eight image pairs that we selected as the most appropriate for UGV applications, the techniques computed disparities for as much as 87% of the points. Although the missing points (and mistakes in the reported matches) could cause problems for vehicle navigation, this level of completeness is an indication that there is a solid basis for building a passive ranging system for an outdoor vehicle.

The number of gross errors varied considerably from image pair to image pair. For most “realistic” images the number was relatively small, ranging from a few “spike” errors to small regions of mistakes. We estimate that for these images there were between 1 and 5% gross errors in the results. In many cases, the worst errors cluster into areas that are “breaking up” for one reason or another (usually poor information plus a poor “guess” for the disparity range); if we can “fix” these areas, then the remaining “spike” errors should be amenable to culling techniques. In any case, most of these errors would have to be eliminated in order for the data to be used directly for planning navigable routes.

The techniques made different mistakes, most of which could be explained by their correlation patch size, search technique, or match verification technique. However, since they made different mistakes, there is a possibility of combining them in a way to check each other and fill in missing data.

All the techniques could be improved significantly with a relatively small amount of effort. This was the first test of this type, requiring the analysis of a large dataset, and it uncovered some weaknesses that can be corrected. One area to be considered is the development of preanalysis

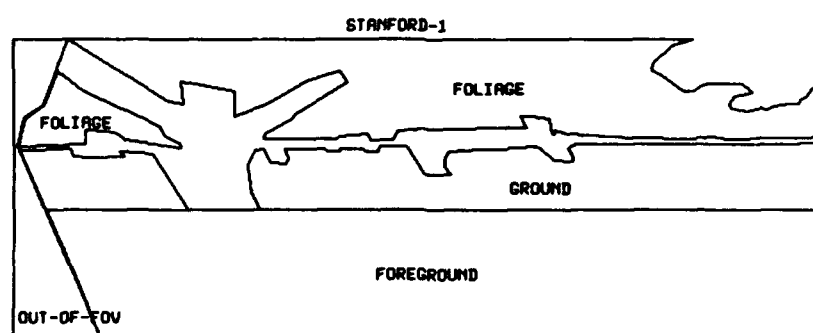
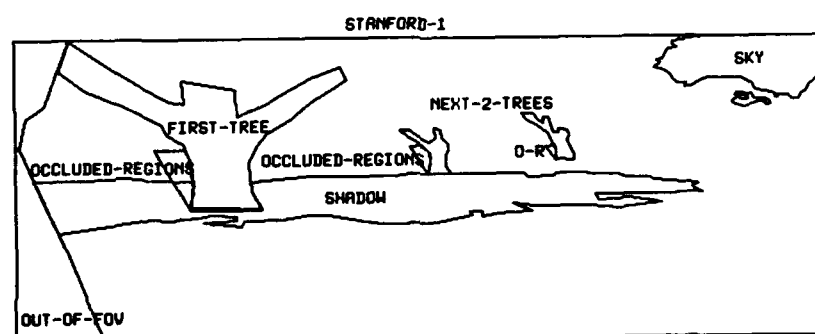


Figure 2: Special-interest regions for the STANFORD image pair from SRI.

	JPL(5)	INRIA(8)	SRI(15)	CMU(9)	Teleos(7)	Weighted Average
INRIA-1 63	66	42	89	35		57
(INRIA-2) (92)	(75)	(60)	(70)	(50)		(67)
SRI-2 94	74	61	64	39		64
Teleos-1 95	81	45	87	77		71
Average 84	74	49	80	50		64

Figure 3: Percentage of "matchable" pixels assigned disparities on all 44 image pairs.

	Arroyo	EPI16	HMMWV1	HMMWV2	J1	Road	Rock	StanDbl	Average
INRIA-1	90	60	67	79	72	40	37	47	62
(INRIA-2)		(85)	(95)	(95)		(90)	(88)	(76)	(86)
SRI-2	91	72	94	94	73	97	94	78	87
Teleos-1	98	72	93	91	74	98	95	72	87
Average	93	68	85	88	73	78	75	66	79

Figure 4: Percentage of "matchable" pixels assigned disparities on the eight most representative pairs.

	Arroyo			HMMWV1			HMMWV2		Rock		
	Bush1	Bush2	Rock	LMound	Rock	RMound	Rock	Etc	LBush	RBush	Rock
Pixels:	56	68	10	130	18	106	26	839	174	105	23
INRIA-1	95	88	100	98	100	88	100	96	67	30	57
(INRIA-2)				(100)	(100)	(100)	(100)	(98)	(74)	(74)	(100)
SRI-2	91	82	90	99	94	96	96	95	68	64	96
Teleos-1	90	100	90	88	100	97	88	94	74	72	57
Average	92	90	93	95	98	94	95	95	70	55	70

	J1			StanDbl		Ball2	Unweighted Average
	RRock	FRock	LRock	1Tree	2&3T	Tennis-Ball	
Pixels:	70	70	98	276	37	145	
INRIA-1	100	100	100	63	86	92	85
(INRIA-2)				(92)	(100)	(94)	(95)
SRI-2	100	99	99	50	76	90	87
Teleos-1	98	89	100	94	62	86	87
Average	99	96	100	69	75	89	86

Figure 5: Percentage of "matchable" pixels on large obstacles assigned disparities.

	Stanford		StanDbl		Unweighted Average
	Shadow	1stTree	Shadow	1stTree	
Pixels:	215	140	448	276	
INRIA-1	40	71	38	63	53
(INRIA-2)	(84)	(96)	(73)	(92)	(86)
SRI-2	59	61	65	50	59
Teleos-1	1	82	29	94	52
Average	33	71	44	69	55

Figure 6: Percentage of "matchable" pixels in shadows assigned disparities.

	iRoad2	J1	Ball2	Ball4	Unweighted
	Road	Bland	Matchable-	Matchable-	Average
Pixels:	1077	229	3126	3126	
INRIA-1	12	22	56	39	32
(INRIA-2)	(86)		(72)	(62)	(61)
SRI-2	63	32	76	43	54
Teleos-1	83	10	86	84	66
Average	53	21	73	55	51

Figure 7: Percentage of "matchable" pixels in bland areas assigned disparities.

techniques to automatically set key parameters, such as patch size and search areas (as Teleos does). Another place for improvement is in the filtering of the results to eliminate matches that differ significantly from their neighbors (as SRI and INRIA do).

There were a few surprises, such as Teleos's successful solution to one set of image pairs from CMU that includes a carpet with a repetitive pattern on it. Teleos's large patches were able to detect large regions of subtle differences, which led to the correct disparities.

5.1 Technique-Oriented Summaries

No one of these algorithms has completely solved the stereo problem, although all can produce basically usable results on most reasonable imagery. Each has strengths and weaknesses—and very often an algorithm's strength on one dataset is its weakness on another!

INRIA's algorithms assume that the images are in epipolar alignment. This makes their searches more efficient, and keeps matches from wandering off of the epipolar lines (for instance, "climbing" the edges of tree trunks). However, when presented with nonepipolar imagery, INRIA-1 fell apart; INRIA-2 did better, but had a persistent problem, producing rough disparity contours, which are apparently due to the way the pyramid was handled. The low-resolution results were simply zoomed-out using pixel replication. This epipolar line constraint also limits the usefulness of INRIA's algorithms on imagery from nonpinhole cameras.

SRI's algorithm mostly disregards the epipolar constraint. Consequently, it had no particular problems handling nonepipolar imagery. However, it failed to match many of the very smooth tree edges in the EPI sequence, probably because its matches "slid" up the linear sides of the trees.

INRIA's algorithms search the entire width of the epipolar line. This helped them to do well on some datasets, but when the ground texture was ambiguous, their technique tended to return no match because of multiple choices.

SRI's algorithm depends on early matches to "set the context", so that later searches for matches can be confined to the disparities in that neighborhood. When there is enough global texture for the initial matches to give a good sampling of the disparities, this works well, enabling SRI-2 to produce ground plane matches where the others couldn't. However, when lack of foreground detail keeps SRI-2 from having the right initial matches, it fails to match, or finds random mismatches.

Teleos's algorithm uses very large windows dynamically skewed to accommodate tilted planes. This causes it to do well on some ground planes where it was able to disambiguate the pattern through minor variations, but not on others where the ground plane tilt was out of the allowed range of skewing. Of course, these large windows also cause it to have problems with any scene containing depth discontinuities—it either finds no match, or tries to blend the foreground object into the background objects, or widens the foreground object out onto the background. In addition, Teleos-1's scanning heuristic creates some rather peculiar artifacts—extending objects in opposite directions on alternate scan lines. However, its ability to "see" into low-contrast situations is very good.

The Teleos system, with its large correlation windows, also produces smaller range images, because it limits matching to areas where the full correlation patch is within the image. In an active vision system, the sensors could be reoriented to center objects of interest that may initially appear on the boundary of an image.

Both INRIA's and SRI's algorithms use fairly small windows. This removes much of the need for window skewing and warping, although on extremely tipped planes, warping would be helpful. INRIA-1, INRIA-2, and SRI-2 all do better on tilted planes if the information is slightly "fuzzy". These algorithms don't do nearly as well in the presence of man-made ambiguous patterns.

SRI's algorithm tends to leave more holes in the data—low-information places that it refuses to try to match, ambiguous places where it can't backmatch successfully, or error matches that it has detected and removed. This gives the data a "lacey" appearance, and it should probably be followed by an interpolation step, to fill in these problem areas. (The SRI technique is capable of interpolation, but it was not used in this evaluation.) SRI-2 often leaves a nice band of no-matches outlining depth discontinuities, where one doesn't really want separate objects "smoothed" together. SRI-2 also often refuses to match areas like the sky, which technically don't have a match.

None of the algorithms currently distinguishes between good image data and the "null data" areas caused by image digitization, reprojection, and so forth. This can lead to rather peculiar mismatches around these areas of null data. All of the algorithms should add the ability to accept a mask telling what parts of the image not to try to match. Better yet would be a preprocessing step to construct these masks automatically.

It was interesting to see how much better all of the algorithms did on the imagery taken by JPL than on the SRI imagery. A major factor is the unusual aspect ratio of the SRI imagery caused by digitizing individual fields, since the vehicle was moving fast enough to show a significant difference between fields. JPL's imagery was taken while the vehicle was standing still. Other differences that may have contributed include image contrast, epipolar geometry, and look angle (SRI's cameras were looking far forward, whereas JPL's were looking down a bit more). We note that the exchange of imagery can help in algorithm development by avoiding inadvertently "tuning" one's algorithm to one's particular style of imagery.

5.2 Open Research Problems

After examining the results from this dataset, we have selected the following topics for future research in the area of low-level passive range sensing:

1. Filtering out gross errors caused by erroneous matches.
2. Handling the wide dynamic range in intensities common in outdoor imagery, from dark shadowed regions up to specularities off shiny surfaces.
3. Handling the large range in adjacent disparities arising from narrow foreground obstacles.

4. Adjusting algorithm parameters automatically to properly handle different image regions, such as bland areas and texture regions.
5. Detecting multiple matches and selecting the correct one, possibly by analyzing multiple images.
6. Providing validation and confidence estimation mechanisms.
7. Detecting occlusion edges and reporting accurate depths on both sides of them.
8. Detecting and characterizing small- to medium-sized obstacles, such as rocks and bushes.
9. Detecting "negative" obstacles, such as holes and ditches.

Although the JISCT dataset did not include examples of the last two areas, they are clearly important for cross-country navigation.

6 Conclusion

As a result of this phase of our stereo evaluation, we can make a few general observations and develop a few ideas for the project's next phase.

First, the time is right for evaluation. If promising computer vision techniques, such as stereo analysis and road following, are to make the transition from the research laboratory to practical systems, their characteristics will have to be well enough documented that system engineers can understand them and predict their behavior. We view this evaluation as the first tentative step toward developing this type of characterization.

Second, evaluations of this type require a significant effort. To give an idea of what is involved in such an evaluation, SRI did the following: gathered imagery from five groups, converted it into a standard format, designed the experimental procedure, distributed the imagery to the participants, collected the results, converted them into a uniform format (correcting for a few mistakes in the original specifications), developed visualization routines, used these routines to interactively examine all the results, developed statistics gathering routines, applied these routines to the results, wrote the report, and finally distributed the report and copies of everyone's results.

Third, ideally an evaluation of this type should be performed periodically to provide estimates of the relative improvements of the techniques.

6.1 Critique of the JISCT Evaluation

Some things that were done correctly:

- We developed a cooperative attitude among the participants. This was the first time our community

had tried establishing an ongoing evaluation process and we knew that we'd make mistakes. We also knew that the participants have their egos involved in their systems, and we wanted to emphasize the constructive aspects of comparing techniques.

- The experimental procedure was almost right. The idea of distributing a large number of stereo pairs, using some for a training set, freezing the "official" algorithm, and then applying it to 45 test pairs is correct. The large number of pairs virtually forced the groups to implement an automatic technique, which they could apply to any image pair. As a result, there are now four systems around the world that can be easily tested on new imagery.
- The idea of asking for precision estimates, confidence estimates, and annotations was correct. Although no group produced them all, future systems will be expected to because this information is so important for higher-level users of the results.
- The basic idea of sharing data from several groups was good because applying the algorithms to this diverse set of images brought to light several implicit and explicit assumptions and parameters in the algorithms.
- Since any evaluation of this type can only include a limited set of imagery that attempts to cover all possible dimensions, the idea of including several small controlled experiments worked well. For example, the set of images from Teleos explored the ability of the algorithms to handle increasing noise; the SRI EPI sequence tested a range of baselines.

Some things that should be changed:

- The lack of ground truth significantly limited the types of automatic "objective" evaluations possible. Ground truth is expensive, but there is no substitute for assessing quantitative issues.
- For this initial phase we built our dataset primarily from existing data. In the future we need to gather data that is more realistic and appropriate to the task. In particular, for UGV tasks, the data should be from the demonstration sites and include examples of the common "obstacles," such as ruts, bushes, rocks, ditches, and water. Future datasets should also include sequences of images and trinocular data, not just individual pairs.
- The whole process took too long (almost a year). Techniques can change faster than that. To be relevant, the results should be returned within a few months. This turnaround time is more possible now that we have been through the process once and have developed routines for analyzing the data.

- More auxiliary data (e.g., calibration information) should be supplied with the dataset. Some techniques rely on this information to reduce search and set key parameters. Also, it will generally be available in most applications.

6.2 Plans for the Next Evaluation Phase

We plan to include three types of imagery in the next dataset: demonstration-related pairs and sequences, a few image-intensified pairs, and some synthetic pairs that are less artifactual than previous ones. One of our goals for this phase is to explore more rugged off-road scenes, including deep ruts, tall grass, and ditches, so we are including several examples of each in the new dataset. The image-intensified data will provide our first look at applying our techniques to night-vision-type imagery. The synthetic data is formed from real pairs by modifying a set of computed disparities, and then forming a new right image based on these disparities. This data, although still not completely realistic, is significantly better than previous versions and provides complete ground truth.

We plan to distribute the dataset to 10 or 15 research groups for analysis. After debugging the process, we are in a position to open up the evaluation to include a wider group of participants.

Reference

Bolles, R.C., H.H. Baker, and M.J. Hannah, "The "JISCT" Stereo Evaluation," SRI International Report, January 1993.

Reconnaissance, Surveillance and Target Acquisition Research for the Unmanned Ground Vehicle Program

**LTC Erik G. Mettala, Ph.D., Deputy Director and
Oscar Firschein, Image Understanding Program Manager**

**Software and Intelligent Systems Technology
Office
Defense Advanced Research Projects Agency**

Abstract

The Department of Defense Unmanned Ground Vehicle Program is a multi-year effort involving the Services, OSD and DARPA. The objective of the program is to first field a teleoperated unmanned ground vehicle system, followed by preplanned product improvements leading to self-navigating systems performing reconnaissance, surveillance, target acquisition and designation missions. This paper describes the recent DARPA procurement for research in reconnaissance, surveillance, and target acquisition (RSTA) for the Unmanned Ground Vehicle (UGV). The research topics of interest are described, followed by a description of the UGV program structure.

1.0 Background

In recent years, Congress "has been concerned about the direction and composition of the many diverse robotics projects undertaken by the armed services and defense agencies."¹ Congress therefore directed the establishment of a DoD robotics master plan in 1989. The diversity of robotics projects that were described in the 1989 plan led to Congressional request in 1990 to consolidate all of the ground robotics vehicle projects "under OSD policy and program direction."

1. Report 101-132 from the Senate Committee on Appropriations on the Department of Defense Appropriations Bill, 1990. Quotations in this section are from Report 101-132.

In response to this Congressional mandate, previously separate ground vehicle related robotics efforts were consolidated in a single program element, under the direction of the Tactical Warfare Programs (TWP) office of the Director for Defense Research and Engineering (DDR&E). Since 1990, the TWP office has been responsible for reporting on the activities of this program element to Congress, providing direction, allocating appropriated funds to projects, and carefully monitoring the progress of all DoD Unmanned Ground Vehicle activities. The Services and the Defense Advanced Research Projects Agency (DARPA) are responsible for the conduct and daily management of the projects.

1.1 Introduction

This paper describes a recent DARPA solicitation for unclassified research projects in Autonomous Systems Technology, focusing on Image Understanding technology needed by Unmanned Air Vehicles, by the Surrogate Semi-autonomous Vehicle (SSV), and later by the Tactical Unmanned Ground Vehicle (TUGV) reconnaissance, surveillance, and target acquisition (RSTA) function. Both the SSV and the TUGV are currently under development. The RSTA requirements include the detection, tracking, model-based identification and location of military ground vehicles at ranges from 200 m to 3 km. The RSTA system is designed to use a wide field of view (wfov) forward looking infrared (FLIR) sensor, approximately 15 degrees x 8 degrees, for target detection. It will then use a narrow field of view (nfov) Laser Radar (LADAR) to gather three-dimensional information from detected target locations for target identification. Target tracking functions will be performed using either the wfov FLIR or video imagery.

Research topics of interest specifically solicited in the Broad Agency Announcement included (1) natural scene understanding, (2) model-based object recognition in Laser Radar (LADAR) imagery, (3) motion compensation (digital image stabilization), (4) target detection in Forward Looking Infrared (FLIR) imagery with natural scene clutter rejection, and (5) Multi-sensor fusion using FLIR and LADAR identification and accurate target location.

We sought research projects whose results could be integrated and demonstrated in SSV RSTA operational scenarios in 1995. Emphasis was on IU techniques embedded in an end-to-end system, rather than on isolated techniques. The solicitation required that the proposed IU research be more than a concept, resulting in programs that could be directly transferred to a Unix-based workstation such as the Sun or the Silicon Graphics for testing, evaluation, and SSV RSTA system integration. RSTA algorithms dependent on special purpose hardware were specifically excluded from this solicitation.

Two possible avenues for technology transfer to the SSV RSTA system were identified. Model-based object identification technology could be optimally integrated into the SSV RSTA target identification system, a modular model-based object identification system based on C/Unix/X-Windows. Where possible, contractors performing research in other technology areas would use C++, CLOS, and the objects and methods defined in the Image Understanding Environment (IUE) described by Mundy [1]. This target platform/environment is meant to assure technology transfer of the final research results, as well as easy interchange for testing, upgrading, and evaluation within the university, government laboratories, and the contractor community. The SSV program has preliminary plans to do all of its system development in C and C++ and will make available to the community at large critical interface designs as the project progresses.

Close coordination of contractors with the existing SSV/TUGV projects was required as well as a series of short-term assignments (several weeks) of SSV/TUGV team members to their research sites to aid in technology transfer. Bidders were additionally encouraged to plan for short term assignments of their technical personnel to the SSV contractor's site in Denver, CO, to support modification, integration, and evaluations of their systems on the SSV. Contractors were expected to attend and to present results at two of the four annually scheduled UGV Demo II Workshops meetings and to prepare an annual paper describing progress for the DARPA Image Understanding Workshop.

Projected total funding for the IU research is approximately \$7,500,000 over three years. Approximately ten three-year

contracts will result from this solicitation. Industry/research center teaming was encouraged to support the use and extension of existing sensor and scene simulation technology into novel IU research approaches to SSV RSTA problems.

1.2 RSTA Research Areas

Analysis of the RSTA subsystem requirements was conducted to identify technological shortcomings in the existing program. The following potential research areas were identified for the IU tech-base effort:

1.2.1 Natural scene understanding.

The results from this general research area will supply guided search process information to the sensor needed to govern efficient sensor search for threats. In the absence of high resolution elevation and cultural map data, the RSTA system itself must use sensor data to determine which areas of the scene are likely to contain enemy threats. This capability may also support long range navigation planning - the current SSV system uses only map information to perform route planning. This research area includes the following sub-components:

Scene component identification (sky, trees, fields, roads, occlusion ridges) in FLIR and RGB imagery that has been acquired from forward-looking, ground-based sensors.

The use of known and monitored host vehicle motion, feature extraction, and image sequence processing, to create 3-D scene maps of SSV operations areas. For example, host motion may be useful in identifying terrain occlusion boundaries that conceal targets.

The use of active vision. Focus-of-attention processing using FLIR or video imagery of natural scenes for target cueing.

1.2.2 Model-based object recognition.

This research topic supports the SSV target recognition requirements of target vehicle recognition. The SSV model-based target recognition approach has a modular design which allows the incorporation of research approaches (for indexing, feature matching, belief propagation, etc.) developed in the IU community. IU research could focus on complete recognition approaches or sub-component technology. Research topics in this area include:

Recognition of tactical targets in ladar imagery. A particular challenge is object recognition in sub-optimal imagery: lower resolution, noisy, or attenuated by atmospheric conditions.

The recognition of occluded targets in ladar imagery. This includes the model-based recognition of whole targets via the recognition of distinctive unoccluded sub-parts.

Ladar sensor, target and scene modeling. Algorithms for simulating target and background characteristics, geometries, and probabilities are key elements of the SSV model-based target identification system.

1.2.3 Motion compensation: digital image stabilization

This research is for techniques that use processing to maintain a stable view of the world from a moving platform. Such approaches must exceed the current technology base that rely on large frame to frame scene overlap. Additionally, approaches in this area must utilize general purpose scalable parallel computation, as opposed to dedicated hardware for this single function.

1.2.4 Target detection in FLIR imagery with natural scene clutter rejection

Open research issues in this field include the use of object, terrain, sensor and atmospheric transmission models to predict target/background contrast at the operational site, and the use of such data to select optimal target detection module parameters.

1.2.5 Multi-sensor fusion using FLIR and LADAR identification.

Research here investigates the cooperative combination of approaches of image segmentation from infrared images and depth maps from LADAR to facilitate object recognition.

1.2.6 LADAR/FLIR Multisensor Fusion.

LADAR/FLIR multisensor fusion research deals with model-based target identification processes that use multiple sensor information to increase algorithm performance. Emphasis will be placed on the combination of information from FLIR and LADAR imagery to increase classification performance.

1.3 UGV/TUGV Program Structure

The technology associated with unmanned systems¹ is maturing faster than the concepts of employment are being developed. The structure of the DoD Unmanned Ground Vehicle Program reflects this reality, in a coordinated evaluation and development program with the objective of first fielding of an unmanned system by 1998.

The TUGV is the principal effort of the current UGV advanced development program. The TUGV program is being planned and managed with an awareness that it represents an initial step in the evolution and fielding of UGVs for combat applications and that its success or failure may have far-reaching consequences.

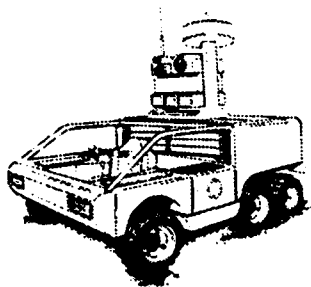
Three principal foci encompass the DoD UGV program. First, several Surrogate Teleoperated Vehicles (STVs) will be developed and used to support Early User Test and Evaluation (EUTE) of UGV concepts. Second, a full scale Engineering and Manufacturing Development (EMD) program will develop the first fielded Teleoperated Unmanned Ground Vehicle. Third, the Unmanned Ground Vehicles Technology Enhancement and Exploitation (UGVTEE) program will focus on maturing those robotics technologies of particular interest to UGV systems. The UGVTEE program is a demonstration-directed effort, including Demo-I, whose principal aims are to mature and transition near-term technology, and Demo-II, whose goal is to develop semi-autonomous navigation technology.

2.0 The Surrogate Teleoperated Vehicle Program

The STV program, managed by the Joint Unmanned Ground Vehicles Office (JUGVO) at the U.S. Army Missile Command (MICOM) in Huntsville, AL will develop 14 Surrogate Teleoperated Vehicles (STVs). These will be used to conduct Early User Test and Evaluation (EUTE), by placing six STVs in a USMC infantry brigade, and six in a U.S. Army brigade for a period of one year, starting in 1992.

1. The term UGV is used in a general sense to include a range of applications. The term Tactical Unmanned Ground Vehicle (TUGV) will refer to a specific project, which is developing one class of UGVs.

FIGURE 1. The Surrogate Teleoperated Vehicle



The STV is a six-wheel-drive, fully amphibious platform. It contains all automotive and navigational components, including sensors and control for teleoperated driving under day, night, and adverse environmental conditions. The platform is powered by a hybrid 25 horsepower (HP) diesel engine and a 3 hp electric motor, for silent locomotion when required. The STV will be able to traverse roads at 35 miles per hour (mph) and travel off-road at 25 mph. Its remote driving speeds will depend: 1) on the skill of the operator in teleoperated mode or, 2) on the sophistication of the software as semi-autonomous capabilities are added. By placing these teleoperated vehicles into the hands of soldiers and marines, the JUGVO will acquire direct access to employment concepts created by users in tactical environments.

3.0 TUGV Engineering and Manufacturing Development

In the Engineering and Manufacturing phase, the selected system contractor(s) will be responsible for fabricating a production-ready TUGV. The Government will conduct Developmental Test and Evaluation and Initial Operational Test and Evaluation of the contractor-provided TUGV prototypes. These tests will determine readiness for production of a first generation TUGV. Milestone III is planned for the end of 1997.

4.0 UGV Technology Enhancement and Exploitation

UGVTEE consists of technology base efforts supporting current and future UGV projects, and involve participation from academe, industry, DoD, DoE and NASA laboratories. The UGVTEE program is directed to exploit robotics advances and mature those technologies that are critical to the robotization of UGV systems. The near-term focus of this program is on providing the mission capabilities and technological enhancement required for

the TUGV. This part of the program (Demo-I) will conclude in FY1992. The long term focus is on image understanding, planning and control technologies that will enhance operational capability and survivability. This part of the program (DEMO-II) has been initiated in FY1991, with the main focus of developing autonomous navigation under battlefield conditions. Technology development will include support for RSTA functions while the UGV is moving, distributed artificial intelligence supporting automated communication with other vehicles, and workload partitioning between vehicles to accomplish mission objectives.

5.0 UGVTEE Demo-I

The principal purpose of Demo-I is to mature critical system component technologies for first generation teleoperated UGVs and demonstrate their readiness for acquisition programs. Based on the results of Demo-I, selected technologies will be integrated into the basic STV for the development of a complete TUGV prototype. The emphasis is on reducing operator work-load while enhancing performance of the RSTA mission.

6.0 UGVTEE Demo-II

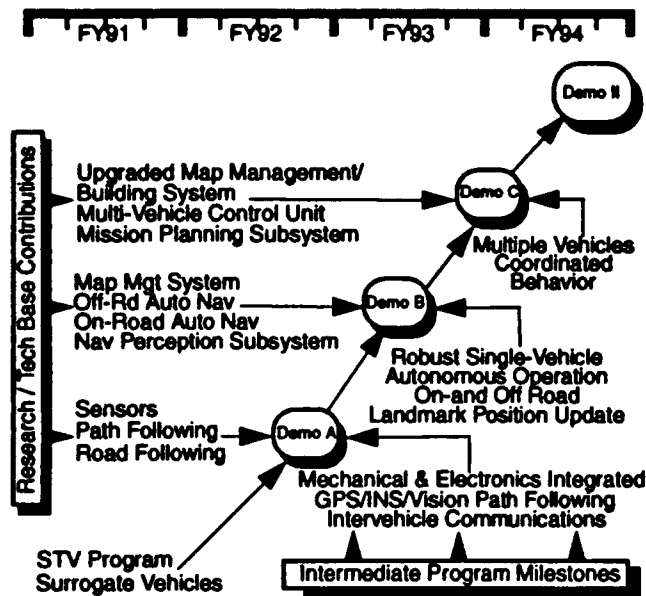
The purpose of Demo-II is to develop and mature those navigation technologies that are critical to evolving UGVs from labor intensive teleoperated systems requiring fibre-optic cables for communication to supervised autonomous systems utilizing low-bandwidth non-line-of-sight communication. The objective of the program will be to demonstrate four semi-autonomous cooperating unmanned ground vehicles performing navigation, reconnaissance, surveillance, target acquisition and target designation.

As shown in Figure 2, Demo-II is a four-phase five year program with three interim demonstrations directed at transitioning research results onto the surrogate vehicles.

6.1 Demo-II Technologies

Realization of the Demo-II objectives will require moderate to substantial increases in capabilities from current state-of-the-art in Image Understanding, Navigation, Planning, Control, and Distributed Artificial Intelligence. The recommendation for approving the Demo-II program was based on research results developed under support by the DARPA Image Understanding¹, Planning², and Robotics Science³ Programs, and others.⁴

FIGURE 2. Demo-II Overview



References

1. Mundy et al, "The Image Understanding Environments Program," Jan 1992 DARPA IU Workshop Proceedings, pp. 185-214, Morgan Kaufmann Publishers, 2929 Campus Drive, San Mateo, CA 94403.

7.0 Conclusion

We have described the recent DARPA procurement for research in RSTA for the UGV. These efforts, to begin in mid-1993, will lead to needed advancements in natural scene understanding, model-based object recognition, motion compensation, target detection in FLIR imagery, and LADAR/FLIR multisensor fusion.

1. Principal Image Understanding Research provided by Carnegie Mellon University [Kanade, Thorpe, Whittaker], University of Massachusetts [Hanson, Riseman, Weems], University of Maryland [Rosenfeld, Davis], University of Pennsylvania [Bajcsy], University of Rochester [Ballard, Brown], Advanced Decision Systems [Morgan], General Electric Corporation [Mundy], SRI [Bolles, Hannah, Strat], Hughes Aerospace Company [Tseng, Nash], Massachusetts Institute of Technology [Poggio, Lozano-Perez], and others.
2. Principal Planning Research provided by University of Maryland [Davis], Carnegie Mellon University [Thorpe, Whittaker], Massachusetts Institute of Technology [Brooks], Advanced Decision Systems [Soldo], University of Michigan [Jain, Durfee], Yale University [Dean], Stanford University [Cannon, LaTombe], and others.
3. Principal Robotics Science Research provided by University of Pennsylvania [Bajcsy], Stanford University [Cannon], Massachusetts Institute of Technology [Brooks, Lozano-Perez], and others.
4. My sincerest apologies to those not listed due to minimal space.

Section V

Image Understanding Environment

The Image Understanding Environment: Overview

J.L. Mundy and IUE Committee*

mundy@crd.ge.com

G.E. Corporate Research and Development
Schenectady, NY 12309

Abstract

The Image Understanding Environment(IUE) project is a five year program, sponsored by DARPA, to develop a common software environment for the development of algorithms and application systems. The ultimate goal of the project is to provide the basic data structures and algorithms which are required to carry out Image Understanding(IU) research and to develop IU applications. This paper provides an overview of the IUE and an update on the status of the project.

Introduction

What is the IUE?

The Image Understanding Environment(IUE) is an object-oriented software system which provides the basic data structures and operations required to implement Image Understanding(IU) algorithms. These data structures and operations are based on the classical mathematical abstractions used in IU research, such as pointsets, transformations and topology. The primary purpose of the Image Understanding Environment(IUE) is to facilitate exchange of research results within the IU community. The IUE will also provide a platform for various demonstrations and tools for DARPA applications. These demonstrations and tools will become a primary channel for IU technology transfer. The IUE will also serve as a conceptual standard for IU data models and algorithms. The availability of standard implementations for algorithms will facilitate performance evaluation of new techniques and to track progress in algorithm improvements. The IUE is designed to support evolution and testing of IU techniques and provide an efficient programming environment for rapid prototyping.

History of the IUE

In late 1989, Rand Waltzman of DARPA, then manager for Image Understanding programs conceived and devel-

oped a new program called I4US. The decoding of this acronym is Intelligent Integrated Interactive Image Understanding. The name has since been shortened to IUE, for Image Understanding Environment.

The IUE program was announced at a meeting for DARPA Principal Investigators in Scottsdale, Arizona at the end of February, 1990. The project goal, as announced by Rand, was a five year program to design and implement a common software environment for the development and demonstration of image understanding algorithms and techniques.

Rand Waltzman convened and chaired three meetings during the 1990-1991 period to develop an consensus in the IU community about the requirements for the IUE. A number of teams were established to suggest specific application scenarios and propose skeleton architectures for the IUE. In April 1991, these team reports were reviewed, and the IUE committee was formed from representatives of each team. In June 1991, Oscar Firschein replaced Rand Waltzman as the IU program manager at DARPA.

Since April 1991, the IUE committee, along with the DARPA IU Program Manager, has met eight times to develop the design and produce a number of documents which specify the design as well as an IU data exchange standard. In September 1992, the committee provided a draft version of the IUE design to DARPA to provide the basis of a solicitation to select the IUE contractor. On January 8th 1993, the IUE BAA was published in the Commerce Business Daily by the contracting agent, the Topographic Engineering Center.

The basic structure of the IUE is captured by a partial summary of the class hierarchy shown in Figure 1. The figure illustrates the central idea of the IUE design and its relationship to mathematical concepts. The following sections provide a brief summary of the IUE design.

Image

The IUE image object class supports many forms of image data, from intensity images to color images to complex composites such as pyramids. IUE images fall into one of two subclasses: simple or composite. Simple images have two primary dimensions, x and y, with possible additional dimensions such as color and/or time.

*The members of the IUE Committee are: Tom Binford-Stanford; Terry Boulton-Columbia; Bob Haralick, V. Ramesh-U. Washington; Al Hanson, Chris Connolly-Umass; Ross Beveridge-Colorado State; Charlie Kohl- AAI; Daryl Lawton-Georgia Tech; Doug Morgan-ADS; Joe Mundy-GE; Keith Price-USC; Tom Strat-SRI

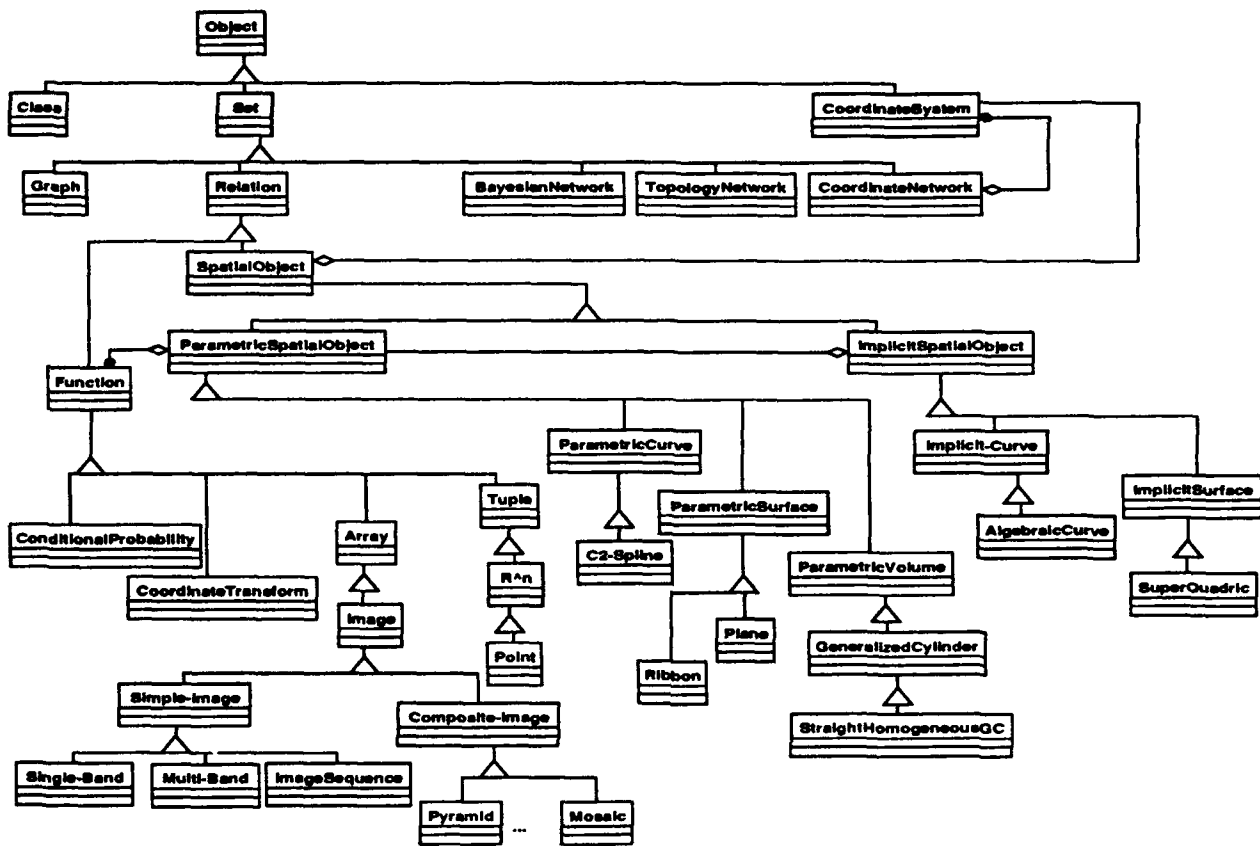


Figure 1: A partial view of the IUE class hierarchy. There are hundreds of class definitions in the current IUE, and this figure represents only a sketch to illustrate the general nature of the hierarchy.

A simple image can be defined as a mapping from $Z \times Z$ to V , where Z is the set of positive and negative integers and V is the set of allowable pixel values:

$$I : Z \times Z \longrightarrow V$$

I may be abstractly viewed as a discrete function of two variables $I(r, c)$ where $r, c \in Z$. For computational reasons it is desirable to restrict the domain of the mapping to a specific subset of $Z \times Z$, usually a rectangular bounded region of the plane. The set of pixels comprising the restricted domain is often called a *region-of-interest* (or *ROI*) and, by extension, the logical image defining the characteristic function is also called an *ROI*. Note that there is no restriction on the connectedness of the pixels in this restricted domain; that is, it is not a true region as defined in region segmentation and may consist of a set of completely isolated pixels. In the IUE, there are two ROIs which affect the domain. While both are called ROIs, their semantics are slightly different. Every image in the IUE has an ROI associated with it, called the *image-ROI*. This ROI may be explicitly represented as a logical image having the same extents and index set as the original image or it may be implicitly represented as the entire image. In addition, each method has an optional parameter which specifies an ROI, called the *in-ROI*. The set of pixels comprising the pixel domain is obtained by intersecting the image-ROI and in-ROI.

Simple-images can map onto different slices of the same underlying pixel data, thus avoiding redundant copies in many common situations. To illustrate, an RGB image is represented as a 3 dimensional shared-array. Associated with it are three 2 dimensional arrays: slices corresponding to each of the three color planes. These four objects—RGB, red, green, and blue images—present different views of the same underlying data. In addition to abstract high level interfaces to pixel data provided by the image objects, the IUE will provide a raw-data interface to pixel data. Raw-data objects are one dimensional containers for large amounts of numeric data, and they support direct pointer access to numeric pixel data. The IUE must efficiently support large images (say 10Kx10K) such as those commonly occurring in photo-interpretation tasks. Thus, a direct tile-mapping mechanism must be associated with raw-data objects which allows large images to be block mapped into memory on a demand basis; this mechanism must be efficient enough to support smooth scrolling (roaming) and zooming.

Image-types commonly occurring in image understanding include: A *color image* with red, green and blue components, where each component is a simple grayscale image. A *range image* represents a depth image acquired from laser triangulation or a time-of-flight range finder. An *image sequence* can be viewed as a queue of images indexed by time. For simple-image-sequences queue length is fixed when the sequence is created and all elements must be simple-images of the same size. In contrast, composite image sequences may contain images of any type and may dynamically grow and contract.

The class of composite-images is very broad and the semantics vary considerably between specializations.

Composite-images provide the flexibility required to develop objects such as image pyramids and mosaics. An *Image-Pyramid* is an ordered set of images, each a power of two reduction of the predecessor. This makes several restrictions evident. The first image must be square, the dimension being a power of 2, i.e., 2^n . The depth is of the pyramid, or number of images, is simply $n + 1$. An image of size $2^n \times 2^n$ is said to reside at level n of the pyramid. A *Mosaic-Image* is a patchwork of images, possibly overlapping, and partially covering an extended 2D area. The proper metaphor is a stack of photographs on a table. Hence, the get-pixel method returns the pixel value for the first image in the set which is defined at the specified point.

Spatial Object

A key element of the IUE design is the *spatial-object* which has the mathematical properties of a pointset in \mathbb{R}^n . The fundamental structures are organized along classical notions of intrinsic dimension, i.e., point, curve, surface and volume. Further distinction is made between implicit and parametric entities. Implicit structures are defined by the vanishing of systems of equations, usually polynomials. Implicit forms are useful for determining incidence or containment. Implicit forms of curves such as the line and conic are used throughout IU research. An example of a commonly used implicit surface is the superquadric.

Parametric structures involve a mapping from a set of parameters to \mathbb{R}^n . The parametric mapping function is a particular type of relation which defines a mapping between two sets of n -tuples, the *Domain* and the *Range*. We further restrict the mapping to be order preserving and one to one. With these properties we can always find a unique point in the domain for a given point in the range and the natural dimension and neighborhood properties are preserved. This is a much stronger condition than is usually associated with the idea of parametric curves or surfaces. The curves here are perhaps more properly called "well-parametrized," where there is a unique inverse for each point in the range of the curve. A typical example of a parametric curve is the spline. A common parametric surface used in IU is the ribbon¹

A coordinate system is associated with the base spatial-object class in order to maintain a consistent definition of coordinates derived from the equational definition of geometric structures. Also associated with all spatial-objects are a bounding-box and centroid point. These ancillary structures enable efficient processing of distance and intersection operations.

There are many other components of the spatial-object hierarchy which are described more fully elsewhere in these proceedings [Ramesh and Committee, 1993].

Coordinates

The geometric relationship between sensors and scenes, among physical objects, and between pixels and the

¹Only restricted classes of ribbon surfaces satisfy the unique inverse property.

world has been a core component of the science of image understanding since its inception. Nearly every IU system makes use of coordinate systems and transforms either implicitly or explicitly. The multitude of representations that have been devised, some of which incorporate arbitrary conventions, has been a key obstacle precluding the transfer and sharing of code and results. The following are definitions associated with coordinate spaces.

Coordinate System A coordinate space, in the mathematical sense. It is represented in the IUE by an instance of a **coordinate-system** class.

Coordinate The *coordinate(s)* of a point are represented by a vector and implemented as a 1D-array. Coordinates are implicitly associated with a coordinate system.

Coordinate-Transform A specification of a mapping between two coordinate spaces. It is implemented in the IUE by an instance of a **coordinate-transform** class.

A coordinate transform specifies the mapping between two coordinate systems, and is represented in the IUE by an instance of a coordinate transform class. There is an implied directionality to each transform, and any individual transform may or may not be invertible.

The relationships among coordinate systems and transforms form a graph. Coordinates can be mapped between any two coordinate systems by finding a sequence of transforms that connect them in the graph. There is a potential problem when more than one path exists between two coordinate systems. Ideally, all such paths specify the same transform, but the existence of numerical errors, and the need for high-performance approximations preclude the treatment of all such paths as equivalent. In IUE, the first path established between two coordinate systems is treated specially – all others are considered to be *derived* transforms. Whenever a transform is requested from the coordinate transform graph, the *basal* (non-derived) transform is retrieved. Derived transforms can be employed when desired by specifying them explicitly. This policy ensures that coordinate transforms are performed consistently and without introduction of excessive numerical error.

Image Features

Central to any Image Understanding research or application program is the extraction and use of image features. Image features are a part of the general spatial object hierarchy in that they combine both geometric and image signal-theoretic concepts.

Image features provide implementations of methods for extraction, property value computations, display, spatial indexing operations, input, output, grouping, and the various iterators over sets of spatial objects (subsets, all in an area, etc.). Image features are often be used as the basis for the **region-of-interest** in image processing operations and thus must support geometric and topological operations.

There are several reasons why developing image features as spatial objects is crucial. First, there is a nat-

ural correspondence between the sequence of topological constructs, e.g. **vertex**, **edge**, and **face** used for spatial objects, and the descriptions of image features for points and junctions, edges, and regions. Access to this topological representation is especially important for describing composite image features such as linked line segments, adjacencies between regions found in segmentations, and perceptual groups. Second, since image features generally correspond to the projection of three dimensional object models, it is useful to have the same underlying operations and representations used for both of them.

Images features collections can be grouped on a variety of properties such as proximity, alignment, curvature, etc. These grouping operations are supported by various spatial indexing schemes such as K-D Trees, quadrees and the Hough transform.

Image features are discussed in more detail elsewhere in the proceedings [Price and Committee, 1993].

Sensors

Unlike some other aspects of the IUE, sensors are an area of active research where few *de facto* standards exist.

Two important aspects of sensing in the context of IU are the device and the data produced by the device, represented by the classes, **sensor** and **sensor-model** respectively. The class **sensor** is the analog of the physical device and is capable of many operations associated with sensing and the production of sensor-models. The output of a sensing operation is stored in an object of type **sensor-model**. The **sensor-model** not only contains a pointer to the generated data, it has a copy of relevant sensor-parameters and provides methods to reason about the geometry of the sensor mapping, and uncertainty in data locations and measured values. The sensor may interact with a external device (e.g. a frame grabber) to get real data, or may generate synthetic data either by embellishing a stored image with additional (assumed) properties or by rendering in conjunction with a scene object. In addition to getting the data, it is also the sensor's task is to determine, from the various attributes of the sensor (e.g. lens parameters, digitizer parameters, etc.), the attributes of the sensor-model (related to its geometric mappings and its uncertainty measurements).

While one might think of sensors producing only image-like data, the mapping concept on which sensors are based is not restricted to physical transducers of energy. Hence, it naturally extends to include the production of spatial objects. This allows us to define a **geometric-sensor** as something that can have the same "lens" slot as a camera and that uses the same sampling pattern as a camera but that maps a scene with many instances of the class **spatial-object** into a **sensor-model** which contains a collection of instances of the class **spatial-object**. For example, a scene full of polygons might be mapped into a collection of vertices and/or line segments.

The IUE User Interface

The IUE will make extensive use of graphical interaction to support the examination of features and to provide convenient tools for model construction, recognition and etc. These tools will be constructed within a uniform user interface methodology and will allow convenient selection and modification of graphic items.

There are three interfaces for dealing with user interactions with the IUE: The user interface (IUEUI), the IUE graphical user interface (IUEGUI), the programmer interface.

The Image Understanding Environment User Interface (IUEUI) is intended to provide flexible, simple, and powerful tools for exploring data, algorithms, and systems. In addition to general principles of good interface design, there are several important objectives which are specific to image understanding and developing the IUE:

Use existing interface standards The interface should be supported by ongoing and future developments in software environments and graphical user interfaces. The interface components should be built on top of existing and emerging interface packages and interface construction toolkits. This is critical for the long term use of our environment because we can depend on continuous advances in these areas that we will want to take advantage of in terms of capabilities and cost (other issues involved with this are discussed in section on graphics software). A good example is the evolving Open GL standard.

A few, powerful interface classes The same general principles of object oriented design used in the IUE should be applied to the interface: abstraction over common operations to provide a small number of types of interface objects which can be freely combined by a user. The interface should not involve understanding a large numbers of unrelated things.

Consistent interaction with other IUE objects The interface should make it straightforward to manipulate and investigate IUE objects. For example, in displaying a spatial object, a user wants to control all aspects of how the domain and the range are displayed. Another aspect of this is providing intelligent default behavior for interacting with IUE objects, such as setting up appropriate types of browsers for different types of objects.

Control of the display and presentation While intelligent defaults and context-based behavior is essential, a user should always be able to override them and have complete control and flexibility.

Support for sophisticated users Naive users will want support for running tailored applications with several interaction aids such as menus while experienced users who want programmability and significant compression and abbreviation in specifying actions.

To realize these objectives the interface of the IUE is described in terms of three levels. The *Graphics Level* is

the underlying "machine independent" package for basic display and graphic operations which tell the screen what to do. Examples would be X and Postscript. The *Interface Kit Level* involves packages for the creation and rapid prototyping of user interfaces and related tools which are built on top of graphics level software. Examples are such things as InterViews, DevGuide, and TAE. This level also includes the tools found in the selected software development environment such as editors and debuggers. It is important that these all be thoughtfully integrated. It should not feel like starting up completely different processes when moving from the debugger and editors for code development to the display and browsing operations of the interface. The *IUEUI Level* consists of the interface objects specialized for image understanding. This includes such things as object displays, plotting displays, several types of browsers, and structures for describing the interface context. The IUEUI consists of a small set of objects which can be freely combined for very powerful results.

The IUE User Interface is described in these proceedings [Lawton and Committee, 1993].

IUE Process Control

Large grain IU operators typically have complex parameter structures. A significant portion of the time spent in developing IU applications is spent in the exploration of the search space defined by the parameters of the operators. For example, a common type of question that needs to be answered by IU researcher is: "What is the most effective Laplacian radius when performing a Zero Crossing segmentation on aerial images of cities?". A great deal of time and effort is spent in determining appropriate values for parameters of a particular operator in a particular domain. Once these parameter values have been determined, it becomes natural to think of the parameterized operator as a new entity that is different from the unspecialized generic operator. This view leads to the concept of an operator as a Task object that can use inheritance and specialization to represent these parameter structures.

IU research also involves a very large amount of processing and data generation; in this type of environment it becomes important to be able to examine the processing history. The Task class readily supports the maintenance of a processing history through the explicit representation of Task parameters. A Task instance can exist in one of three states depending on the specification of the input and output parameters for the Task: it may either be partially specified, fully specified, or completed. In this way, Task instances describe both the complete input and output specification and the processing status of Tasks. The Task objects thus provide a complete description of the large granularity image understanding processing that has occurred in a user environment.

Associated Classes

Another aspect of large grain IU processes is that they are used by the IU researcher as a set of tools within an experimental toolbox. Researchers require a flexible mechanism for control and data chaining that al-

lows them to construct experiments that combine individual Tasks into more complex algorithms. The Task class hierarchy provides this mechanism through the *CompoundTask* and *DataflowGraph* object classes. With these classes, the user may chain individual Task objects together, either through programs or through the use of an interactive graphical interface.

DataflowGraphs allow the user to specify data pathways between Task objects. The Tasks in a DataflowGraph can have some subset of their input parameters specified dynamically; the input values of the dynamic parameters are specified by the values of output parameters of other Task objects. Whenever values have been specified for all required input parameters, the Task object is executed. Other DataflowGraph objects, such as DataflowConditional nodes and DataGenerator nodes, provide the control constructs that make the DataflowGraph an effective programming tool. A DataflowGraph may be constructed either in C, LISP, or at the interface level, to form these complex processes. In C or Lisp, this complex Task control can be also implemented through a message passing paradigm in which Task instances are parameterized and controlled through messages (generic function calls) from a controlling program.

Applications

The *Tasks* that will be supported by the IUE will cover a wide range of algorithms and tools. It will be expected that the set of Tasks that are included with the IUE will expand rapidly as the IUE begins to receive wide use and the research groups using the system begin to contribute their own research tools. The following are examples of TaskGroups specified by the IUE design.

ImageProcessing Tools that map image data to image data.

ImageSegmentation Tools that map image data to symbolic data.

PerceptualOrganization Grouping tools to map symbolic data to symbolic data.

GeometricFitting Tools that fit geometric entities to symbolic data.

ObjectMatching Tools mapping object descriptions to symbolic data.

ModelConstruction Tools for creation and manipulation of object descriptions.

The Future of the IUE

The IUE is planned to be developed in three phases: basic, core and version 1.0. The basic version of the IUE accounts for the elementary classes and methods required to support development of core IU classes and algorithms. The core IUE represents a useful intersection of current practice in IU research. The core will provide representations for the major structures and methods used to implement IU algorithms, such as segmentation, grouping, matching and modeling. Finally, version 1.0 will consist of the core plus selected support for curves and curved surfaces as well as demonstration algorithms which illustrate the use of the core class library.

Currently, the IUE committee is developing a data exchange standard which will be immediately useful for exchanging research results. The data exchange format is based on the IUE class hierarchy and provides an ASCII representation for the construction of class instances. The syntax is Lisp-like and can be easily parsed by LEX/YACC. The initial goal of the data exchange specification is to support technology transfer within the RADIUS project. The format is discussed in more detail in these proceedings [Mundy *et al.*, 1993].

At this writing, the solicitation process is underway to select the IUE integrating contractor. It is expected that the selection will be made in the first half of 1993. The implementation of the basic and core versions of the IUE are expected to take about two years from start of contract with version 1.0 being released at the end of the project.

During these project phases there will be continuous review and consultation to insure that the IUE meets the requirements of the IU community. Any suggestions or comments concerning design or implementation issues are welcome and may be directed to mundy@crd.ge.com.

References

- [Lawton and Committee, 1993] D. Lawton and IUE Committee. The image understanding environment user interface. In *Proc. DARPA Image Understanding Workshop*, 1993.
- [Mundy *et al.*, 1993] J.L. Mundy, R. Welty, and IUE Committee. The image understanding environment: Data exchange. In *Proc. DARPA Image Understanding Workshop*, 1993.
- [Price and Committee, 1993] K. Price and IUE Committee. The image understanding environment: image features. In *Proc. DARPA Image Understanding Workshop*, 1993.
- [Ramesh and Committee, 1993] V. Ramesh and IUE Committee. Spatial objects in the image understanding environment. In *Proc. DARPA Image Understanding Workshop*, 1993.

The IUE User Interface*

Daryl T. Lawton, David Dai, MaryAnn Frogge, Warren F. Gardner,
Heather Pritchett, Andrew T. Rathkopf, Ian Smith and the IUE Committee†

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

Abstract

We describe the design and initial prototyping of the user interface of the DARPA Image Understanding Environment (IUE) and tools for documentation, tutorials, and publication that will facilitate the use and adoption of the IUE.

1 Introduction

The user interface of the Image Understanding Environment (IUE) is intended to provide flexible, simple, and powerful tools for exploring data, algorithms, and systems [Mundy and others, 1992]. The general principles of object oriented design used in developing the IUE object hierarchy and programming constructs have also been applied to the interface: abstraction over common operations to provide a small number of interface objects which can be freely combined by a user. The interface has been designed to have a consistent interaction with IUE objects and their semantics, especially the abstraction in the IUE object hierarchy. Thus, the display and browsing operations are sensitive to the class similarities for objects such as images, image registered features, and spatial objects. Using and becoming comfortable with the interface hopefully will not involve understanding a large numbers of unrelated things.

An equally important part of the user interface is what it does not develop. The IUE user interface must leverage extensively off of existing (and emerging) interface and graphics packages and standards. The interface needs to be supported by ongoing and future developments in software environments and graphical user interfaces. This is critical for the long term use of the IUE. We can depend on continuous advances in all these

areas that the IUE will need to take advantage of in terms of capabilities and cost.

To realize this, the interface is being developed in terms of three levels. The **Graphics Level** is the underlying "machine independent" package for display and graphic operations which tell the screen what to do. Examples would be X, GL, OpenGL, and Phigs. The **Interface Support Level** involves packages for the creation and rapid prototyping of user interfaces and related tools which are built on top of graphics level software. This also includes the tools found in the selected software development environment such as editors and debuggers. The **Image Understanding Environment User Interface (IUEUI) Level** consists of the interface objects specialized for image understanding. This includes such things as object displays, plotting displays, several types of browsers, and structures for describing the interface context. The IUEUI consists of a small set of objects which can be freely combined. The specifications of these objects is relatively independent of the other two levels although much of the current prototyping activities are directed towards understanding how to best realize the functionality of the IUEUI objects with respect to these two levels, especially for accessibility and limiting the eventual cost of the IUE for users.

The basic functional components of the IUE interface are:

- **Displays:** These deal with mapping spatial objects and images (or sets of spatial objects and images) onto two-dimensional display windows. There are types of displays for displaying images and image registered features, for plotting functional relations between attributes and components of spatial objects; and for displaying surfaces.
- **Browsers:** These deal with presenting textual and symbolic information about objects. There are different types of browsers for operations such as inspecting the values in a spatial object, for performing interactive queries with respect to databases and sets of objects, and for inspecting relational graphs and networks.
- **Interface Context Descriptors:** These are for describing the state of the interface and interface objects. Examples are such things as the current color-look-up table for a given display; the current

*This research is supported by the Advanced Research Projects Agency of the Department of Defense and is monitored by the U. S. Army Topographic Engineering Center under contract No. DACA76-92-C-0016

†The members of the IUE Committee are: Tom Binford-Stanford; Terry Boult-Columbia; Bob Haralick, V. Ramesh-U. Washington; Al Hanson, Chris Connolly-Umass; Ross Beveridge-Colorado State; Charlie Kohl-AII; Daryl T. Lawton-Georgia Tech; Doug Morgan-ADS; Joe Mundy-GE; Keith Price-USC; Tom Strat-SRI

display window or browser; and links between interface objects which describe related views. This information supports intelligent default behaviors.

- **Command Language and Command Buffer:** A user can control his interaction with objects using an interactive command language. The commands can be used in code and to create scripts. This also provides a complete description of the functionality of the user interface.
- **Simplified, programmable access to Graphical User Interface (GUI) objects:** This is intended to provide programmer access to several of the objects commonly found in Graphical User Interface Construction Kits such as knobs, sliders, text buffers, menus. These can then be used in applications and to extend the interface

With the exception of GUI objects, we now look at each of these in more detail. Our focus here is on the functional components of the user interface and their attributes.

2 Object Displays

Object Displays are for viewing and interacting with objects by mapping them onto two-dimensional display windows. This involves nearly all IUE objects: images, curves, regions, object models, surfaces, vector fields, etc. Object displays involve a wide range of actions such as displaying an image and image registered features; displaying networks of objects such as stereo images, multi-resolution pyramids, image sequences (and this can involve having several linked windows for the different images; cycling through displays of the different components; or mapping the different components onto different planes of the display buffer and combining the images through transparency or color addition); display of models and predicted segmentations as overlays; and interactively inspecting and manipulating displayed objects and applying operations to them.

There is a strong relation between spatial objects and displays. Most IUE objects are expressed as relations between sets. In displaying an object, values from one set are used to specify a position in a display window and the corresponding values from another set are used to specify an attribute such as intensity, color, overlay, transparency effect, etc., that is displayed at the position. A basic example is an image, where one set is described by the indices of the coordinate system of the image and the other by the color or intensity values associated with the particular image coordinate. A discrete curve is a mapping from integer indices onto two-dimensional positions with respect to an image coordinate system. Displaying a curve as an overlay on top of an image, involves mapping two-dimensional positions along the curve onto window positions using the same transformation that was used for the display of the image. The color/intensity of the display at these points can be based upon registered values associated with the curve (such as approximated curvature). For example, a user might want to display an intensity image in 8-bits of grey-level intensity and then overlay extracted curves

on top of this with the display of curvature values along the curve mapped onto 8-bits of red intensity.

We refer to the operations that are involved with specifying the mapping onto a screen position the **position methods**. These involve operations such as panning, zooming, perspective transformations, and warping operations. The machinery for this comes directly from the coordinate system transformation methods. The operations that involve mapping onto a particular window value are called the **value methods** and these involve such things as setting up the CLUT, point-mapping functions (functions applied to the value at an object position prior to display), transparency effects, etc.

The basic processing flow for displays is shown in figure 1. Displays take place with respect to a context which involves such things as the current object, the current description of the transformation from an object to the display window, the current color look-up table, links between IUE objects, and several other attributes. Several display operations involve setting up these contextual attributes. Displaying an object, such as an image or image registered features, involves iterating over the object and applying the specified position methods to determine the position in a window at which to generate a display and also applying the specified value methods. In interactive processing, a selection operation is performed with respect to the display context to return a value at a selected location. Graphics are also done with respect to the display context (The processing flow in figure 1 is idealized in several respects. Many display operations don't involve iterating over an object but manipulating the color look up table and display buffer directly. Rendered objects or displays which involve warping or interpolation are more naturally expressed as iterating over the display window itself. Displays can also involve a discrete sampling of other objects using square pixel neighborhoods).

The object display methods are organized into the following classes:

Value Methods: These are methods that control how values in the specified object(s) get mapped onto screen attributes such as color and intensity; how to configure planes in the screen buffer for the display of color images; how many panes to use for overlays; particular functions and conditions to apply to object values prior to display. This includes operations such as overlays, mapping onto different color bands, transparency, and others.

Position Methods: These are methods that control how positions in specified object(s) get mapped onto a display window. This includes operations such as panning, zooming, perspective views, and some types of warping.

Display Window Attributes: These involve controlling attributes of the display window such as position, size, attributes of the title bar, event handling for the mouse; and resizing operations.

Link Methods: For linking different displays (and browsers and GUI widgets). Examples are window to window zooming, display of stereo and pyramidal

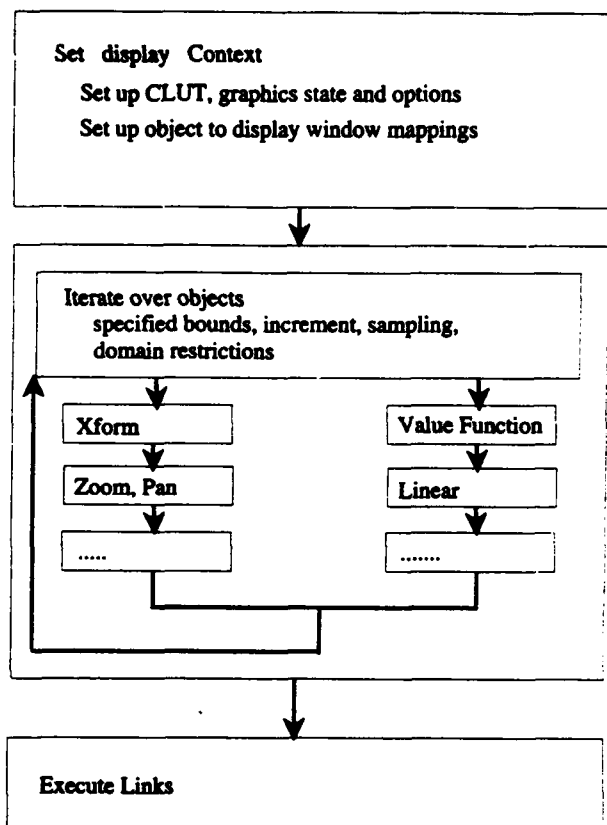


Figure 1: Basic Processing Flow for Displays

images in multiple images. This involves creating links and specifying the operations and transformations associated with links between interface objects

Interaction Methods: These involve interaction and manipulation of displayed object(s) in a display. Operations include object selection, recovering object positions and values from a mouse click, applying functions to selected objects

Graphics Methods: Display registered graphics for drawing lines, text, three-dimensional objects, etc.

History Methods: Many objects can be mapped to the same display window. These are methods to coordinate displays over time, such as cycling through an image sequence, playing an animation of displays.

Archiving Methods: Printing an object display, writing an object display to file, writing an object display to video tape

2.1 Value Methods and the CLUT

Value methods are used to specify how values in an object are mapped onto display window attributes such

as color and intensity. There are several types of methods for this. **CLUT-segment methods** involve setting up a color look up table (CLUT). They involve creating named segments and associating some number of bits with each segment, such as specifying 8 bits for red, 8 bits for green, and 8 bits for blue. **CLUT-value methods** involve associating color and intensity values with CLUT indices. These operations can be applied to CLUT-segments. For example, the red component of the color look up table can be mapped onto red values in several different ways: by linear interpolation between specified shades of red or by a spline through specified color values. **Overlay methods** involve setting up overlay planes. Overlay planes can be displayed and cleared separately of underlying intensity displays. **Object-mapping methods** deal with taking object values and mapping them onto color table indices. For example, the CLUT-segment for red intensity could be set for a linearly interpolated 255 shades of red but the actual object values in an object could range from values such as -1000 to 2000. The **Linear** object-mapping method specifies to linearly interpolate from this range of object-values to the available shades of red. There can be a linear mapping from the object onto color table indices, but the color table may be set up for a non-linear mapping onto actual intensities displayed on the screen. **Value-function methods** are user specified functions that are applied to specified object values to map them onto CLUT indices. Examples are conditional expressions that determine what value to map a particular object value onto. In Lisp this is straightforward. In C and C++, it involves a run-time interpreter which we want to be of as minimal complexity as possible. Other value methods deal with transparency, blinking, and logical operations on bitplanes.

2.2 Position Methods

Position methods are used to specify mappings from spatial objects and images onto two dimensional display windows. They specify where to display something in a display window. For example, position methods are used to specify pan, zoom, and scaling parameters to relate an image to a display window.

The position methods and transformation networks used by the interface are defined by the coordinate-transforms used in the IUE. The interface generally requires transforms of images and image registered objects onto display windows and simple types of interpolation and sampling. More complicated mappings, such as image warping and the generation of rendered objects for a specific sensor use methods from the sensor and scene classes and image packages for warping. These are either used to generate an image which is then displayed or the object specific display methods can be invoked through the interface.

The coordinate transformations and networks are very important for the interaction methods. In this case, the user indicates some position in the display window and then the mapping from the object to the display window is inverted so that the corresponding values and position in the object can be determined and accessed.

The object display is able to do this for images and invertible geometric transformations. For others, such as interacting with a potentially complex, rendered solid, methods from the other classes, such as the sensor and scene class, are needed. It is also possible that there are other representations of a rendered scene, such as an image-registered depth map that contains pointers to all the surfaces that project to a given pixel, ordered by depth that can be generated to simplify the interactive processing.

There is often hardware supported pan and zoom for images that should be accessible through the position methods, even though this is machine dependent.

2.3 Interaction Methods

The **Interaction Methods** are for interacting with and inspecting objects through the context associated with a display, such as the current object to display transformation. The methods associated with this are built on top of the event-handling mechanisms of the supporting environment. Interacting with an object through a display involves using the position mapping from the object to the window. This is straightforward if the mapping is invertible and there is no interpolation or warping. This is usually the case for images and image registered objects. It can also involve geometric intersection using the ray of projection corresponding to a selected window position. For other objects, such as closed analytical surfaces, the reverse mapping is more complicated and involves general spatial object methods that need to be accessed by the interaction methods.

The current object to be interacted with can be explicitly specified or selected. Selection can require disambiguation if there are multiple overlapping objects or complex spatial objects. The user may be required to use a spatial index image (an image of pointers to objects which occupy a given position) or use geometrical data base operations in the IUE. Both are potentially expensive and don't reflect operations specific to the interface but are general IUE spatial data base operations that need to be invoked through the interface to return the selected object(s) and object position from the object to display mapping.

There can be a variety of interaction devices (minimally it should specify a screen location), but we are assuming a mouse with at least two distinguished buttons and text-input from the keyboard. In the interactive mode:

- The current position of the mouse is stored in variables for the (*mouse-x, mouse-y*) of the current display window. Associated with this is the corresponding positions (*current-position*) and values (*current-values*) in the specified objects. The default is to only deal with geometry only to inverting coordinate system transforms and not sampling or interpolation relative to the objects. Since several objects can be interacted with at the same time, these lists will consist of lists of positions and values.
- When the mouse is clicked, the values for the window position, the corresponding object positions

and values are stored in separate lists. Interactively selected Functions can then use items in these lists as parameters.

- A user can associate functions and command-sequences with keys and mouse-states so the functions can be called interactively. The functions are stored in a table index by mouse-state or keyboard-event. The functions can be a sequence of specified interface commands and can be interactively applied to the values in the different lists. Functions are selected using keyboard input (numbers) or mouse-state (mouse-down, mouse-up, mouse-drag for the left, right, and, if it is available, the middle mouse button). Function selection may also be based upon a count of the number of mouse-clicks for specified mouse button.
- There may be a default spatial index associated with a display window. This is memory intensive but can help with a lot of the interactive operations, especially the selection operations.

2.4 Graphics

Often a user will want to perform graphical displays of text, two dimensional graphics, and three dimensional graphics. Examples are annotating a display, indicating where some action is occurring (the position of an epipolar line, translational flow paths, etc.), projecting a wire-frame of a model onto an image. Much of this functionality will come directly from an existing graphics packages that the IUE will utilize. The graphic displays need to take place in three different modes:

- they can occur in the coordinate system of the display window. In this case displays only occur with respect to the window coordinate system.
- they can occur in the coordinate system of a displayed object, such as drawing a line with respect to the inverse mapping from window to object coordinates
- generating corresponding IUE objects. Thus, in drawing a line in image coordinates, an corresponding instance of an IUE line object would be produced. When the wire-frame model is displayed, each line-segment and junction would be created as an object in the IUE. This is very useful for producing data for testing routines. This mode can be coupled with the interactive processing mode to allow for the interaction creation of data. This maybe restricted to relatively simple objects such as polygons, curves, and so forth.

2.5 Types of Object Displays

We have distinguished four types of object displays:

- The **image or pixel display** is for viewing images and image registered features.
- The **local graphics display** displays objects by mapping their values onto parameterized graphic objects such as lines and cubes. Examples are displaying vector fields and edgels.

- The surface display is for displaying objects that get mapped onto mesh or rendered surfaces.
- The plot display is for displaying functional relations between objects. Examples are one-dimensional, two-dimensional, three-dimensional graphs, histograms, scatter grams, and views of functions and tables.

These different types are distinguished by specific methods but all inherit a large number of similar methods from the general display class. For example, overlay operations are similar for a surface display and for an image display, although they can look quite different (In one case it appear like drawing in solid colors in image registered coordinates on top of a displayed image and in the other it would be rendering the colors onto a displayed surface).

2.5.1 Local Graphics

Local Graphic Displays are a subclass of object displays which map object values onto parameterized graphics, such as a line, a square, a perspective view of a cube, Chernoff faces, or a user specified function. A common example is a vector display which will map each component from a pair of image onto the x and y components of a vector. Using the general display methods, the vectors can be displayed as an overlay on top of an image or through indices in a CLUT. For visualizing three dimensional attributes in register across an image, the user can display unit cubes with their orientation computed from the specified components of the display object. The graphic display can be a piece of graphics code which will be positioned to the projected location of the pixel.

There will be specialized local graphic displays for vectors and different types of edges because of their heavy usage. It will be possible to display the horizontal and vertical edgels in the cracks between pixels or to place a single edge at the center of a pixel with it's orientation determined by the specified components objects

2.5.2 Plot Displays

There are several different types of plot displays: one-dimensional, two-dimensional, three-dimensional graphs, histograms, scattergrams, perspective views of functions and others. Examples of plot displays can be found in several data visualization packages and mathematics packages such as Mathematica and GNUPlot [Wolfram, 1991]. In using such packages in the IUE, it is important to bear in mind cost limitations on bundled software and potential problems with data compatibility and speed. Plots also need to be compatible with the general display methods for such things as

- mouse interaction methods: for selecting a position in a graph and then having access to the domain point and the range point. An example is interactive segmentation from a histogram displayed as a plot
- links between plot displays and other types of displays

- most of the view transformations for such things as scaling and zooming
- overlaying plots in different colors

We are currently exploring the use of the plotting capabilities in GNUPlot to be used for use in the IUE. It is essentially free and all the source code is available.

3 Browsers

Browsers are used for interacting with text based or symbolic descriptions of objects. They are used for actions such as queries over set of objects, determining and inspecting relationships between objects, process monitoring, and inspecting values in an object. The browser and browser-related classes are being designed so they can readily be built on top of existing interface construction kits.

There are two general types of browsers: **Field-Browsers** and **Graph-Browsers** of which only field browsers are currently being implemented. Field browsers are built from component objects which are found in several GUIs:

- A field appears as a rectangular box which can be filled with text, icons, colors, colored text, text in particular fonts, or user-specified graphics. Fields can have actions associated with them when they are selected or a user changes the values in them.
- Fields can be organized into connected horizontal or vertical field groups where each field as a unique index in the Field Group. The fields in field groups will generally have different objects displayed in them. An example comes from the object registered browser where a field group can correspond to a display of registered values from different objects. For better visualization, these can be displayed in different colors, fonts, etc, in addition to their position in the field group. A field group can also have a distinct boundary
- Field Groups can be organized into field matrices where in each group as a unique index set in the field matrix. Objects and sets of objects can be mapped onto the matrix.
- Field Matrices can be scrollable as a way to control the mapping of an object (or object set) onto the Field Matrix

We distinguish between four types of field browsers which inherit from the general Field browser class:

- **Object-Registered Browser:** This contains values extracted from a spatial object, such as the intensity values in some square neighborhood of an image. Depending on the dimensionality of the object (or relationships between component objects), this can be presented as a one-dimensional array, a two-dimensional Array, or multiple two-dimensional arrays and describe curves, images, image sequences, pyramids.
- **Set/Database Browser:** This is presented as an array of fields. Each row of fields corresponds to

selected attributes of a particular object and each column corresponds to common attributes over the set (or database) of objects. An example would be browsing the database which describes the current active objects in the IUE to find the most recently created image from some operations.

- **Object Attribute Browser:** Each row corresponds to the value of an attribute for an object. This is used for inspecting a single object.
- **Hierarchical Browser:** Useful for text based inspection of graph structures and trees. When an item is selected, the related items (along some relational dimension) are displayed in the next column.

The methods associated with browsers are very similar to those with displays, suggesting a more general IUE Interface object class. The position methods for browsers involve how an object (or set of objects) gets mapped onto the fields of a browser. For object registered browsers, these are essentially the same as with displays (see figure 2). The fields are analogous to pixels in a display window, although they can be filled with textual information. For DataBase browsers the position methods specify how objects are mapped onto rows of the browser and how attributes are mapped onto columns (See figure 4). The position methods for mapping from graphs and networks onto a hierarchical browsers involve keeping track of different paths through a networks and nodes and arcs that have been traversed. Browsers can also be linked to browsers, displays, and user specified interface widgets. The following examples have been implemented using the FORMS GUI kit on SGIs [Overmars, 1991].

3.1 Object Registered Browser

The object registered browser is used to inspect the values in a neighborhood of a spatial object. A common example is inspecting the image values about a selected point. It is very much like the display of a spatial object in a display window, but instead of the values being mapped onto window positions and screen intensities and colors, values are mapped onto field locations and general field attributes such as colored text in specified fonts, colors, and icons. The attributes of and the specification of the mapping from an object onto an object registered browser is shown in figure 2. A set of spatial objects are mapped onto a matrix of fields in the object registered browser. This mapping involves several parts: a coordinate transform from the N-dimensional spatial objects to the two-dimensional object registered browser; the type of interpolation to be performed if this mapping doesn't involve discrete values; what to do when browsing beyond the boundaries of the spatial object. Also shown is a navigation tool to interactively access position methods to position the object registered browser with respect to a set of spatial objects. The browser is linked to a display window which shows the position of the browser with respect to the bounding rectangular prism of the spatial object. This display would be updated when the browser is moved with respect to the spatial objects. Figure 3 shows an implemented two-

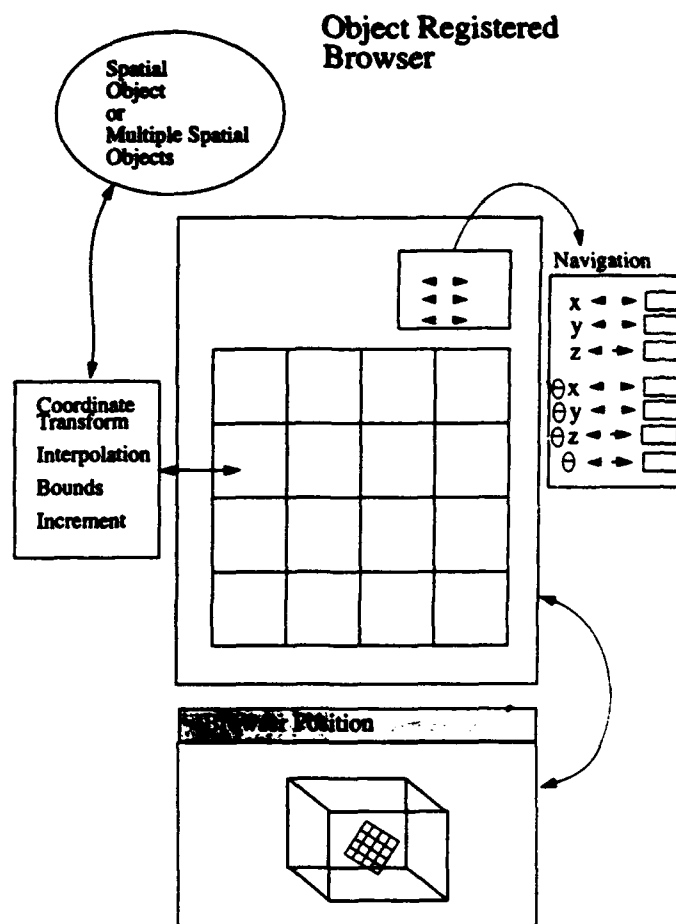


Figure 2: Object Registered Browser

dimensional Object Registered Browser in which is displayed two images and the computed difference of the two images, each in separate fields. Each image is displayed in a different color and the field containing the difference image uses the background color to encode the magnitude of the difference.

3.2 Set/Database Browser

The set/database browser is for inspecting the attributes of a set of objects. It enables interactive queries can be performed via the browser. This is especially useful for keeping track of instances of objects (an object selected in a Set/Database browser should probably default to the current-object so it could be displayed immediately). There are two structures used for describing the mapping from a database onto the browser. One is the set of selected attributes which correspond to the columns. The other is the current set of items which satisfy a query and the indices into the first and last element of this set which are displayed in the corresponding rows of the browser (see figure 4). Figure 5 shows an example using the Set/Database browser to inspect a set of line segments and then to sort them by slope.

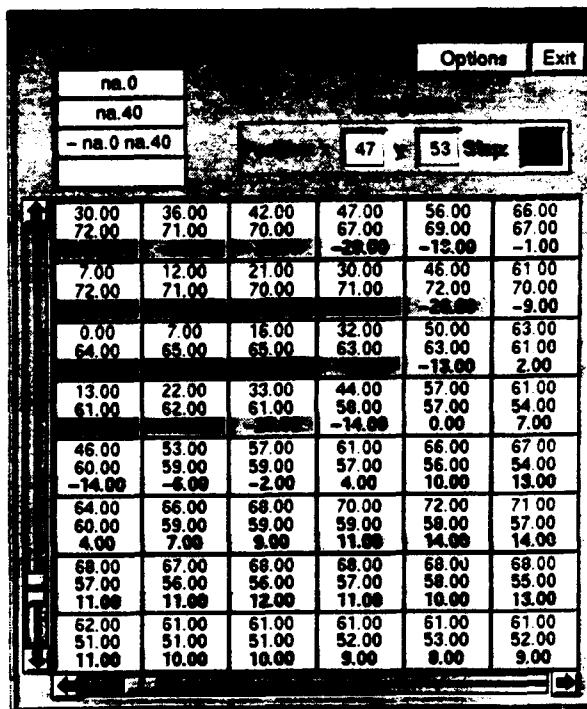


Figure 3: Object Registered Browser applied to an image

3.3 Object Attribute Browser

An **Object Attribute Browser** is for inspecting the attributes of an object or several object with the same types of attributes. It uses an ordered list of object attributes to determine which attributes of the object to display. Figure 6 shows an example of a Object Attribute Browser applied to the attributes of an image object.

3.4 Hierarchical Browser

A **hierarchical-browser** is for inspecting graphs and network objects. Instead of one large field-matrix, it consists of linked Nx1 field-matrices. Each column corresponds to a set of nodes. When a node is selected, the types of relations (arcs) are displayed in the *current - arc - browser*. When a type of arc is selected, the nodes with that type of relation are displayed in the adjacent (right) column. Several structures are used to describe (and update) the mapping from the graph onto the successive browser columns. The current node is the most recently selected node. The current path is stored as well as the nodes that have been visited. Figure 7 shows a hierarchical browser applied to an instance of a polyhedral mesh.

4 Interface Context

There are several data structures for describing the context of the interface. These are used for intelligent defaulting and for saving the state of the interface. These include:

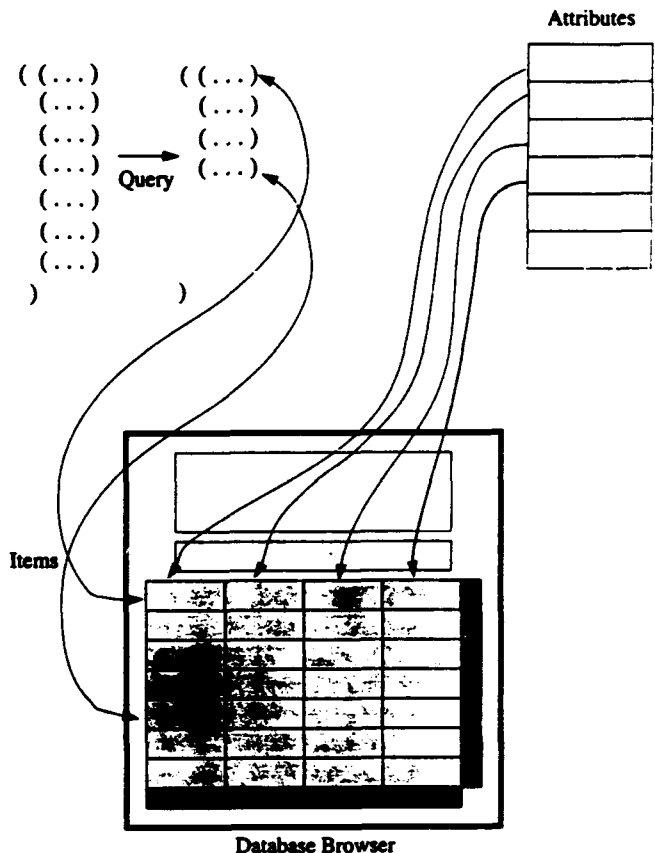


Figure 4: Set/DataBase browser

- **Object-Display-Mapping:** Structures which describe the mapping from an object onto a display. This includes the viewing transformation between an object and a display window, the value-mapping of how the object is displayed and a reference to a particular CLUT.
- **Object-Browsing-Mapping:** Structures which describes the mapping from an object(s) or database onto a browser
- **Display Context:** Structures which describes the current context for a display for such things as the current window, the current object, the current object display mapping, the current display command, the current mouse-selected object position and value, the lists of interactively selected object values and positions. For example, if neither a display or an object is specified, it will default to the most recently used.
- **Browse Context:** Related structures for browsers. Such things as the current browser, the current data base, query history, and others.
- **History:** The sequence of display or browsing actions for a particular window or browser are saved

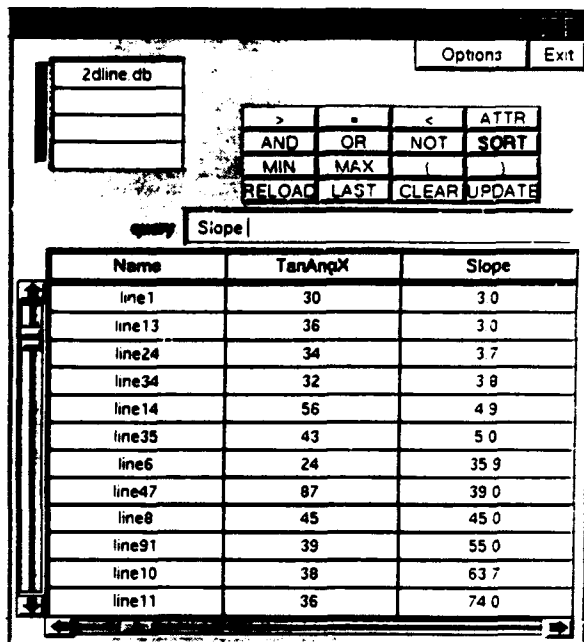


Figure 5: Set/DataBase browser applied to set of line segments from Data Exchange Format

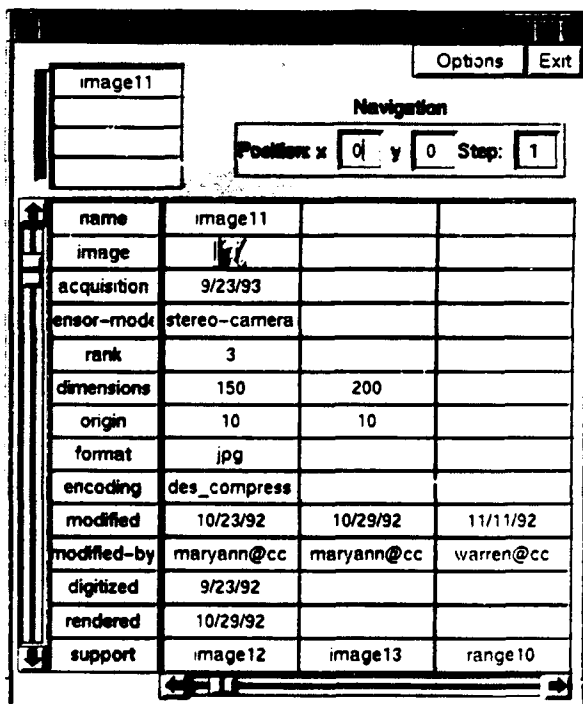


Figure 6: Object Attribute Browser applied to an image

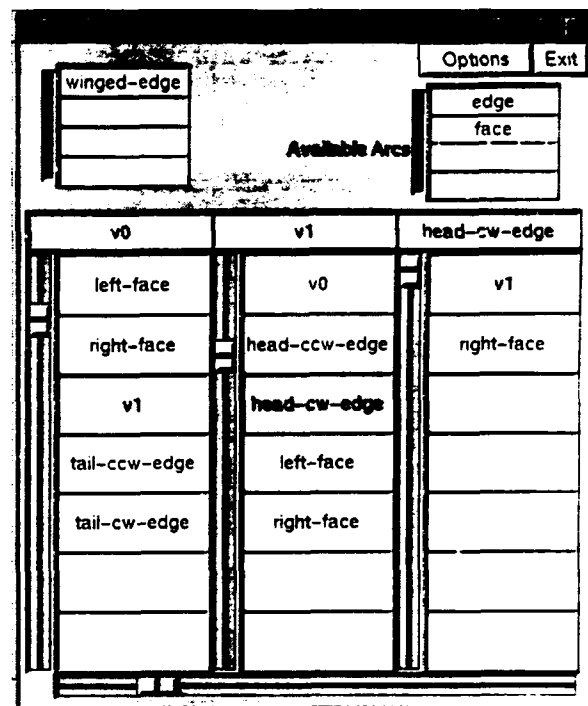


Figure 7: Hierarchical Browser applied to a description of a polyhedral mesh from the Data Exchange Format

and can be reaccessed and used for creating animations. In addition which objects have been displayed or browsers is also stored.

- **Default layouts for windows and browsers:** The desired layout of windows and browsers can be saved and be available to a user when he starts using the IUE. Users may prefer different interfaces (arrangement and instantiation of the basic interface objects) depending on the task or level of sophistication.
- **Object Display Links:** A structure which describes the concatenation of a display or browsing operation between IUE interface objects.

The context description is an extension to the underlying context usually provided by the graphics level. It should be possible to read and save context descriptions.

4.1 Links

Links support operations such as window to window zooming, displaying the same object from different views or using different value-mappings, and controlling displays using interactive widgets like sliders and knobs. Linked displays are useful for displaying composite data such as stereo image pairs or pyramids. When something happens in a parent display (or browser), another display will perform an action using information from the parent display. The action can be a display operation operation or executing a sequence of commands associated with the link, such as a set of commands from the interactive command language.

The mapping between a spatial object and a display window in one window can be concatenated with the display specification in another. A common example is using one window to zoom onto the display in another or using one window to display a selected portion of another (Panning and Zooming are so common they will be directly supported via an interactive tool).

We have specified constraints on links to avoid many complex and pathological things that can happen. Linked displays and browsers are only updated when a display action is performed, not when changes are made to the displayed object. Individual links are bi-directional, but no cycles are allowed in the graph formed from all of the links between IUE interface objects.

5 Command Language

Users will be able to specify all interface actions through an interactive command language and be able to access all the functionality of the interface. Display operations can be performed interactively through the command buffer. The command language will have intelligent defaults and abbreviations (such as displaying to the current window if none is specified). In addition, the commands will be usable in non-interactive code for creating scripts and general display routines. All of the functionality of the interface is accessible through an interactive command language which encompasses the overall functionality of the interface.

A concern with the interface command language is that it becomes another language that people will need to memorize. This is not an issue for development in Lisp since the display operations can be called interactively like any other function, but it is a significant issue with C++. We intend for the command language to be as simple as possible, with a limited syntax. Most arguments are specified via keywords and correspond directly to interface object methods. There are also defaults for command specification. And the IUE will probably eventually support intelligent prompting to complete the commands. The general syntax is

*IUE-interface-object object-set [keyword arguments]**

For example,

```
[*w1* image1 :p]
```

means to display image1 using a pixel-type display in window *w1* using the current display context associated with the display in window *w1*. The brackets are used to indicate separate commands. If the last display operation was of type :p in *w1*, then only:

```
[image1]
```

needs to be specified. More detailed examples are presented below.

An important operation for displaying spatial objects is the ability to apply functions to objects prior to displaying or interacting with them. These operations almost always don't involve creating a new object. An example is manipulating the underlying color look up table to perform a thresholding operation. In this case, there is no thresholded image object produced, only what is displayed in a window. This goes by many names in different systems such as Pixel Mapping Functions, Dynamic Color, Generalized Color Look-Up Tables.

There are two aspects to such functions. First, there are limitations on the types of functions that should be specified for application to an object when it is displayed. Operations such as zooming, panning, manipulating the color look up table, specifying which planes in the display buffer are used, and simple point-wise algebra with limited conditional evaluation, are very useful and will be supported. But, it doesn't make sense for operations such as generalized warping or detailed processing over a neighborhood or generalized intersection to be done by via interface commands. Second, there are also language specific aspects for specifying function application to objects prior to display. In Lisp, it is straightforward to pass lambda expression or closures which are applied to each position or value prior to display. In C, this requires a library of standard functions and an interpreter.

In the actual operation of the IUE, it is not necessary that all interactions take place through this command language: some will be invoked by menus and special keys and refer to the current display context. An important part of the design of the IUE interface entails how commands (and which commands) are mapped onto menus and other interactive devices. This is especially important since the interface will support a wide community of users ranging from naive users who are interacting with hardened applications to developers. Naive users may want lots of interactive devices such as specialized menus while experienced users will want more powerful, abbreviated commands. Advanced users will become very adept at shortcuts that should be provided.

5.1 Examples

The following presents some examples of specified display operations using the command language.

```
[*w1* image1
:p
:linear 0 128 *screen-min* *screen-max*]
```

This would display to window *w1* using the current defaults. The range of object values from 0 to 128 are linearly mapped onto the range of values *screen-min* and *screen-max*.

```
[image :p]
[edge-image :overlay red]
```

An image is displayed in the current window using a pixel-type display. The edge image is then overlayed on

top of this. Wherever the edge-image is equal to 0 nothing is displayed in the red overlay plane and wherever the edge-image is equal to 1, the corresponding pixel is set in the red overlay plane.

```
[overlay-colors (red, green, blue, violet)]
[image :p :value-function
  (if (image.value > 10) red blue)]
```

The first command tells the current display to use the specified overlay colors. The second will display red in the overlay plane at a screen pixel corresponding to an image pixel if the image value is greater than 10, otherwise it will display blue.

```
[spatialIndexImage
:p
:value-function
  (if (label-image.value = NULL)
    0
    (length (spatialIndexImage.value)))
:linear 0 20 0 *screen-max*]
```

This function displays a spatial index image (an image where each pixel contains a list of all the objects which occupy that pixel). The value function determines the number of objects in this list and the linear function maps this onto available screen intensities.

```
[image1 image2
:p
:value-function
  (image1.val - image2.val)
:linear -20 20 *min* *max*]
```

to display the difference between two images. Other common value functions would be for type conversion and display histogram transforms. The user can also specify functions in the interactive mode to be applied to the values in the different queues. For example:

```
[image
:i
:1 [p :overlay-plane clear]
  [p image :value-function
    (if (image.value >
      object-values[1])
      red blue)]
```

The user has selected an image location with a mouse click and the corresponding queues have been filled with the window and object positions and values. Thereafter, when the user hits the terminal key 1, the overlay planes will be cleared and all image locations with a value greater than the value at the selected image location will be displayed in red, otherwise blue, in the overlay planes. *image.value* is a dummy variable that refers to the current value in image which is being displayed. *object-values[1]* refers to the value selected using

a mouse click in the display window and stored in the object-value queue. *red blue* refers to globally defined overlay colors. Recall that the *:value function* specifies the operation to be applied to an object value to map it onto a screen intensity or color.

```
[link *w1* :zoom 2 2 :pan 50 50]
```

This links **w1** to the current window and concatenates a zoom and a pan transformation.

```
[RegionDB
:p
:positions RegionDB.locations
:values RegionDB.textureMeasure
:linear 0 100 *min* *max*
:red-8]
```

This says to display the RegionDB in the current display window with the positions coming from the locations attribute of the regions in the regionsDB and the values by taking the Region DB texture mappings and using a linear mapping from these onto screen intensities in 8 bits of red.

```
[*W1* histogram :plot1d ]
[*W2* image :p]
[*W1* histogram
:i
:1 [min = object-values[1].x]
  [max = object-values[2].x]
  [*W2* image
  :p
  :value-function
    (if ((image.value > min) &&
      (image.value < max))
      blue red)]
```

This is an example of plotting used for interactive histogram segmentation in which the interaction methods lets us click on the axis of a plotted function to returns the x coordinate and the y-value of the displayed object and then use these values to specify peaks in a histogram. Here the user has plotted a histogram in **W1**. He then selects the range of values by clicking on the displayed histogram. The current-object-value contains the x and y value from the displayed histogram. These are stored in the local values min and max. When the user hits the key 1, the selected range of values are displayed in the blue overlay plane in **W2**.

6 Additional Features

Even though our focus has been on developing the core functionality of the user interface, there are several other features that have been considered for use with the interface. Some of these can be built on top of the interface objects and operations described previously. These are important candidates as packages and libraries to augment the core IUE.

One important area involves interactive task management tools. Examples can be found in the data-flow editor in the Cantata portion of Khoros and the Task editor in KBVision. Another area that several people feel is important is developing graph browsers for the display of graphs and networks, generally representing object or values as nodes and using links to describe relations. Graph browsers can have difficulties when trying to display several nodes with arbitrary relations between them in that the connections between the nodes can begin to obscure the overall display. A typical use would be for the display of a constraint or coordinate transform network.

There are probably hundreds of nice interactive controls for displays and visualization that exist in different environments, such as interactively manipulating the object-value to screen-intensity function by interactively shaping a function; selecting color look-up tables; modifying color look-up tables; interactively building display commands using templates or command browsers; floating tool palettes of interactive drawing tools; etc. In general, such tools can be very useful, but it is extremely important that there be a consistent look and feel with different applications that are based on the IUE. This will be partially achieved by depending on the underlying graphical user interface to supply the basic interface objects.

Other useful interface tools are:

- **Interactive Selection and Modification of color lookup tables and display mapping functions;** cycling through different color look up tables
- **A dialog box for setting up system defaults and initializing characteristics of the IUE:** initial layout, font selected, level of expertise, etc.
- **Access to and Integrated use of Established Visualization Packages:** There are several data visualization products and it would be nice to have a modular interface to these
- **Mensuration tools:** Such things as rulers, grid overlays, and the use of multiple cursors mark of distances and points of reference. These probably can be built on top of basic interface capabilities and the display of IUE objects (in particular, the display interaction methods and IUE objects such as bit-mapped regions, line-objects).
- **Interactive Object Creation (Draw Objects):** It should be possible to create object interactively. This is useful for creating idealized data for testing and development. This should be supported by the display interaction methods and access to the instantiation methods associated with spatial objects
- **Incorporating Hardware Accelerators:** So the interface and the IUE in general can modularly incorporate different hardware accelerators.
- **Display Buffer Optimization:** The display buffer itself is a short term memory for manipulating the view of a displayed object. A useful feature would be routines to directly access the display buffer or

performing specific display operations in ways optimized for particular types of displays.

7 Status - Implementation Trade-offs

We are currently prototyping many different parts of the user interface to complete the functional specification and to answer basic implementation questions about choices regarding GUIs and user interface toolkits. This will help to simplify the job of the eventual integrating contractor. For reasons of rapid development, current implementations are taking place in C and C++ on Silicon Graphics machines using the GL graphics library, Motif, and the FORMS user interface toolkit. We have been able to put up the general display object and the different browsers and hope to use these as initial browsers and displays specialized for use with the Data Exchange Format. We are exploring extensions to GNUPlot so it is compatible with the methods associated with the general display class and can provide an inexpensive plotting package. We are also evaluating OPENGL as a possible machine independent graphics package.

8 User Facilitation Tools

The IUE will be supported by on-line documentation and tutorials. The tools for implementing these will also be available for enhanced communication and publication by scientists and developers who use the IUE. While there is significant activity in developing documentation and hypermedia toolkits, they remain largely machine dependent with no clear standardization. We are developing a simple documentation tool called Knowledge Weasel (KW) which is based on Lucid Emacs 19 and existing media editing tools.

Knowledge Weasel (KW) is a presentation and authoring system designed to support annotation using several different types of media. A simple analogy for KW is reading a book or attending a lecture and being able to make diverse types of comments and annotations on the material. In reality, such unrestricted annotations and comments made with respect to real books and lectures could create a significant mess (especially if made by several different people), so in developing KW we have extended this simple metaphor in several ways. The first is to provide a general format for annotations that can include several different types of media. An annotation is a common record structure wrapped around instances of different types of media such as text files, sound, drawings, postscript files, GNU-plots, code running in the GDB debugger, and others. Annotations are implemented much as a property lists in LISP with attributes and values and are displayed as buttons with an associated region of support. When an annotation is selected it performs an operation specific to the type of annotation selected. Annotations are created using existing media editing tools for operations such as recording a sound, drawing packages, calls to other branched processes, grabbing a portion of the screen. The second extension has been to develop different types of navigation, organization and presentation tools to keep users from being overwhelmed with a great deal of possibly

irrelevant information. Users can prune the set of annotations that they want to deal with and also how they are displayed. Annotations are structured to make possible intelligent processing, perhaps eventually including rule-based processing for automatic presentation and "ferreting" of information (hence the name).

We are implementing KW on Lucid Emacs 19 which is in turn based on the X window system. Lucid's implementation of Emacs Lisp provides primitives for handling display attributes such as windows, fonts, and colors. Lucid Emacs version 19 has a built-in lisp interpreter for Emacs Lisp and this lisp variant provides a wide variety of primitives that are useful for manipulating text, processes, and/or files. It is available via anonymous FTP on the Internet, and is also the basis of a commercial product. Knowledge Weasel is chiefly written in Emacs Lisp but some parts, such as the part which interacts with the operating system's lock daemon (lockd), is in C and communicates via pipes with the Emacs Lisp portion of the implementation.

Figure 8 shows an example using some of the current features of KW. A user is reading some text about histogram equalization from a text file in Emacs. He has selected some annotations for display (these could be comments from other users or references to other materials). One annotation corresponds to bringing up the corresponding code and then executing it step-by-step in the GNU debugger. One nice thing about the integration with GNU-tools and Emacs is that it is possible to directly annotate running programs for step by step commentary. The user has selected the button *View Histogram* which is associated with a GNUPlot-type annotation. Annotations are displayed in a larger font of text (which is colored). The actual display of annotations is controlled by a user. Annotations can be conditionally displayed and mapped onto different colors and fonts.

We have begun using an initial version of KW to develop an on-line version of the Low Level Vision course taught at Georgia Tech. We also plan to use it as part of a computer vision algorithms course where students will select a paper from the literature, implement the corresponding algorithms and use KW to develop a tutorial presentation of the paper.

8.1 CD-ROM

A significant instance of technology transfer is the DARPA IU Proceedings and workshop. For the next meeting, we hope to enhance this by having the workshop proceedings available on CD-ROM, and integrated with the Data Exchange Format, a documentation and browsing tool such as Knowledge Weasel, and, possibly, the IUE itself. This will enable an extraordinary type of paper which includes data, code, additional references, animations, and extensive annotations and cross-references.

References

[Mundy and others, 1992] J. Mundy et al. The image understanding environments program. In *Proceedings of the DARPA Image Understanding Workshop*, pages 185-214, 1992. San Diego, CA.

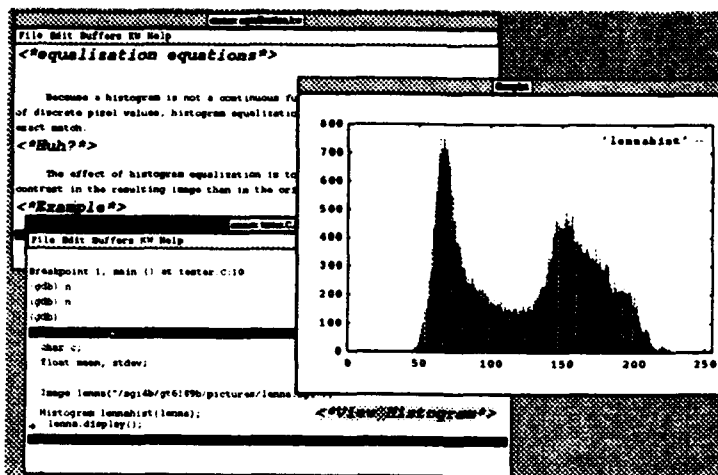


Figure 8: KW Example

[Overmars, 1991] Mark H. Overmars. *Forms Library: A Graphical User Interface Toolkit for Silicon Graphics Workstations*. Utrecht University, 2 edition, December 1991.

[Wolfram, 1991] Stephen Wolfram. *Mathematica: A System for Doing Mathematics by Computer*. Addison-Wesley, Redwood City, CA, 1991.

The Image Understanding Environment: Data Exchange

J.L. Mundy and R. Welty and Keith Price and IUE Committee*

mundy@crd.ge.com

G.E. Corporate Research and Development
Schenectady, NY 12309

Abstract

A major activity of the IUE committee is the design of a data exchange standard for IU algorithm results. The exchange standard is formulated according to object-oriented design principles and is based on the class hierarchy of the IUE specification. This paper provides an overview of the exchange format.

Introduction

As the design of the IUE progressed, it became clear that the concepts for IU data structures and operations could be applied to the formulation of a data exchange standard for IU research and application results. Such a standard is badly needed since two application-oriented programs are now underway at DARPA which involve the cooperation of a large number of research groups. One of these projects, RADIUS¹ involves a number of university and other research institutions who are developing IU algorithms to support site modeling and image analysis. The second project is the Unmanned Ground Vehicle(UGV) project which is focussed on autonomous navigation and reconnaissance. It is clear that both of these projects can benefit from the capability to exchange detailed results of algorithms such as image segmentation, feature grouping and model matching.

The IUE data exchange standard based on an object-oriented representation of the main structures used in IU research and applications. The primary emphasis is on the relationship between image signal data and geometric structures. Much of IU research is involved with grouping and matching of geometry derived from images. Another major area of processing and representation is associated with the derivation of camera parameters associated with camera calibration and camera motion.

The design of the exchange format is based on these design principles: character based formats (ASCII) will

be used for simplicity, object descriptions will be stored in Lisp-like lists, and the format facilitates the transfer of data between different systems.

Character format The first design choice eliminates binary formats, which may be necessary for efficient storage of some objects, but simplifies transportability between different systems. Note that this format is not primarily for the storage of image data, but for the storage of more geometric and relational IUE object data.

LISP-like syntax The second principle implies only that parentheses (or another suitably defined macro characters and reserved words) surround the data. Otherwise the format is relatively free-form. Since Lisp has a simple syntax, this assumption provides a small set of delimiters to break the data and an easy mechanism to read the data in Lisp. For the C++ implementation, the parsing will be straightforward and through the use of a few key words, the format can be efficiently parsed by Lex and Yacc parsing mechanisms.

Free form output By expecting the user to have relative freedom in the output sequencing, we are not required to analyze the data to find relatively efficient formats. The user will specify what objects are to be saved, the order of the objects and the set of object slots that are included. This approach simplifies the output process, but does requires the user to insure that all the required object instances are defined before used by other classes. The design assumes that the format conversion is a single pass operation.

Portability The final principle requires a format that is easily read and written in either Lisp or C, one that is not dependent on the host machine, and one that can be transformed into other internal representations independent of the Image Understanding Environment. The syntax is also very similar to the class construction styles used in both C++ and CLOS, so the action routines of the parser do not require much reconfiguration of the data to form class constructors.

Relation to Other Standards

There are many standards for binary image data file formats, such as NITF, TIFF and even as ASCII such as

*The members of the IUE Committee are: Tom Binford-Stanford; Terry Boult-Columbia; Bob Haralick, V. Ramesh-U. Washington; Al Hanson, Chris Connolly-Umass; Ross Beveridge-Colorado State; Charlie Kohl-AAI; Daryl Lawton-Georgia Tech; Doug Morgan-ADS; Joe Mundy-GE; Keith Price-USC; Tom Strat-SRI

¹Research and Development for Image Understanding Systems

Postscript. The Programmers Image Kernel(PIK) standard provides additional representations for image processing operations as well as representation for various image data types. There are also standards for the representation of CAD geometry such as the Initial Graphics Exchange Specification(IGES). Some aspects of IU are also captured by standards associated with the exchange of geographic data such as the DOD Vector Product Format(VPF) Standard and the Standard Interchange Format(SIF) used to represent simulation database entities. More recently, standards are emerging for the representation of product design information under the Product Data Exchange Using STEP(PDES) program of the Department of Commerce. STEP is an international standard for the representation of product geometry and functionality. In addition, some aspects of product definition and test requirements are being addressed by the DOD Computer-aided Acquisition and Logistic Support(CALS) program.

These existing standards do not cover the breadth of mathematical and physical concepts inherent in IU algorithms. For example, in the case of image segmentation, there are many attributes which must be defined, such as edge strength, or edgel orientation. In order for different research groups to use each other's results these attributes must be defined according to a standard naming convention and associated mathematical definition. As another example, IU algorithms depend on many types of grouping operations, some quite unique to IU, such as the Hough transform and are not supported by other exchange formats.

Finally, IU is a rapidly evolving discipline and it is necessary to have an easily extensible standard and the same time maintain the compatibility of existing data. The IUE object-oriented design approach enables this flexibility through inheritance and class definitions which can be provided in the exchange file itself.

In the remaining sections, we provide a summary of the ideas behind the development of the standard and provide the syntax for the current version of the file format.

Core Exchange Data Structures

The initial scope of the data exchange format is based on the core data structures in the IUE. The following summarizes the classes to be supported in the initial release of the standard.

1. Image Data

- (a) 8 and 16 bit intensity data
- (b) 8bit, 3-channel color data
- (c) multi-channel landsat data
- (d) range with registered intensity data

2. Spatial Object

- (a) basic spatial object
- (b) 2d and 3d pointsets
- (c) 2d and 3d implicit and parametric lines
- (d) 2d and 3d implicit and parametric planes
- (e) 2d polygonal topology(e.g. vertex, edge, face)

- (f) 3d polyhedral topology
- (g) 2d and 3d box neighborhoods

3. Transforms

- (a) 2d and 3d Euclidean
- (b) 3D quaternion
- (c) 4x4 Projective
- (d) 4x3 Image Transform
- (e) UTM and lat-long earth coordinates

4. Spatial Indices

- (a) 2D grid
- (b) 2D quadtree
- (c) 2D r-Tree
- (d) 2D Hough array

5. Image Features

- (a) edgel
- (b) pixel chain
- (c) edgel chain
- (d) segmentation line segment
- (e) connected line segments
- (f) image region

6. Sensors

- (a) perspective camera
- (b) stereo pair
- (c) moving linear array camera
- (d) range camera

These structures represent only a portion of the IUE design, but have been selected as an initial implementation goal and are likely to provide maximum utility to the RADIUS and UGV projects mentioned in the introduction.

An Example

The following example is taken from the IUE specification document which includes examples for data exchange. The specification is mainly concerned with naming and definition of region attributes.

2d-image-region

Description A 2d-image-region is a connected set of image pixels, registered with a set of images by operations such histogram segmentation, region-growing, model surface projection. Sometime attributes and operations involving regions are based upon the set of points which comprise the region (i.e., compactness, Euler number), some operations and attributes are based upon the image values at these locations (i.e., average intensity in the image area corresponding to the region). This concept can be generalized for voxel processing in analogy to the class block.

Superclasses

image-feature 2d-unordered-pointset face
connected-image-pointset

Pseudo Slots(Attributes)

1-chns: list(image-1-chain)

Multiple interior boundaries as a list of usually **image-pixel-chains**. These chains can be 4 connected or 8-connected. The boundary is usually composed of several pixel-chains which intersect at image vertices.

area integer

The size of the region in image pixels a method of face specialized for discrete pixel regions.

number-of-holes: integer

The number of holes in the region.

minimum-bounding-rectangle:

2d-aligned-rectangle-neighborhood

centroid: point

The position of the centroid of the region.

scatter-matrix-of-pixel-positions:

```
vector[2](vector[2](float))
```

Provides covariance of x and y coordinates of pixels in the region.

compactness: float

Ratio of perimeter to area.

adjacent-regions: list(2d-image-region)

A list of regions which share a boundary with self. The shared boundary descriptions are contained in the *inferiors* of each region.

intensity-distribution: vector[2](float)

A distribution (assumed gaussian) with two slots, mean and variance. These are floats with the values computed using all the points in the region and the corresponding intensity image. If the nature of the image is unknown, then this is the mean of its values. For other, known, image types such as red, infra-red, range, etc. other attributes will be used, but they have the general same form.

red-distribution: vector[2](float)

Distribution for the red component.

green-distribution: vector[2](float)

Distribution for the green component.

blue-distribution: vector[2](float)

Distribution for the blue component.

xxx-distribution: vector[2](float)

Distribution for the XXX component. Since these are implemented as pseudoslots, any number of such distributions can be specified according to application requirements. The general name for the different distributions is `<band-name>-distribution` for the variety of image bands.

An example of the data exchange format for region A in the figure.

```
(make 2d-image-region "reg-A"
  (slot 1-chns (list
    (make image-1-chain "ichn-a-b-A"
      (slot edges (list
        (make 2d-pixel-chain "pchn-a-b",
```

```

(slot v0 (make 2d-vertex 'vert-a'
  (slot p (vector 2 integer 11 13))))
(slot v1 (make 2d-vertex 'vert-b'
  (slot p (vector 2 integer 13 9))))
(slot n 24)
(slot chain-code-sequence
  (vector 23 integer 4 4 4 4 3 2
    4 3 1 3 1 2 2 1 0 0 0 0 7 7 7 6 7)
)
)
)
(make 2d-pixel-chain 'pchn-b-a-1'
  (slot v0 (use 'vert-b'))
  (slot v1 (use 'vert-a'))
  (slot n 5)
  (slot chain-code-sequence
    (vector 4 integer 1 1 2 1)
  )
)
))
(slot dir (vector 2 integer 1 1))
(slot closed-p true)
)
(make image-1-chain '1c-c-c-A'
  (slot edges (list
    (make 2d-pixel-chain 'pchn-c-c'
      (slot v0 (make 2d-vertex 'vert-c'
        (slot p (vector 2 integer 7 7))))
      (slot v1 (use 'vert-c'))
      (slot n 14)
      (slot chain-code-sequence
        (vector 13 integer 6 5 7 6 0 0 0
          1 2 2 3 4 4)
      )
    )
  )
)
))
(slot dir (vector 1 integer 1))
(slot closed-p true)
)
))
(slot neighborhood
  (make 2d-image-pixel-neighborhood 'n-5'
    (slot num-nghbrs 7)))
(slot number-of-holes 1)
(slot adjacent-regions (list 'reg-B'))
(slot intensity-distribution
  (vector 2 float 135.2 3.4))
)
)

```

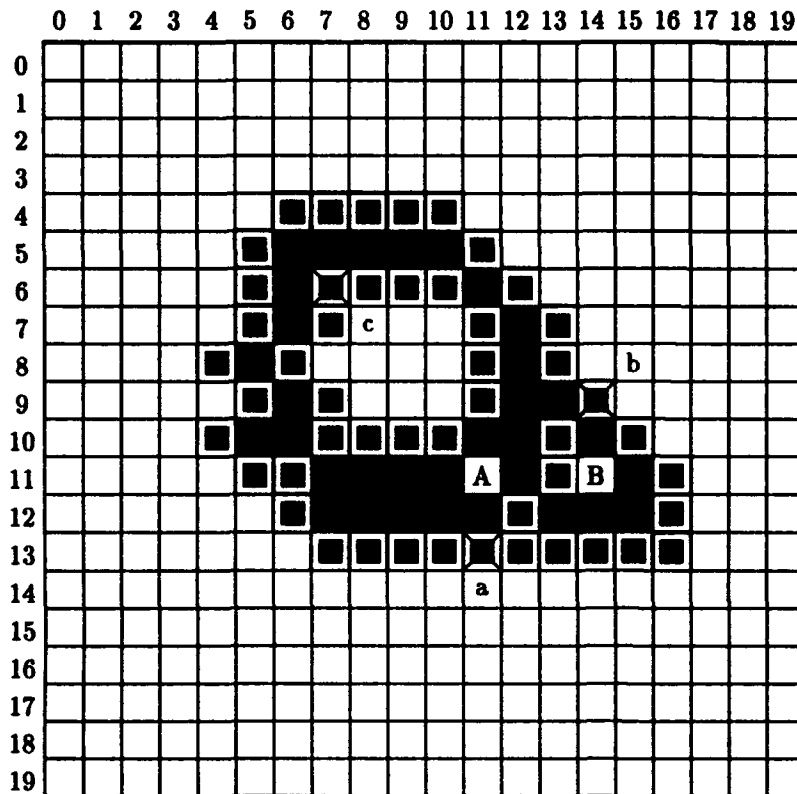
The Exchange Format

Basic syntax

At the most basic level, IUE Exchange Format looks somewhat like a lisp file; the format is designed to be readable by most lisp readers without much difficulty, should this be necessary. A prototype C parser generated by the standard Unix(TM) utilities Lex and Yacc is available upon request.

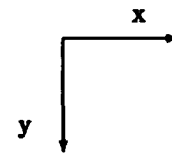
File Organization

A file in IUE Exchange Format consists of an IUE Exchange Format version identifier, followed by a series of






 interior boundary vertex



IUE Class instance descriptions, default slot value settings, and references. The general model is that the content an individual IUE file will correspond to one hierarchy, e.g. one site or one building. IUE files may reference external objects by identifying their IUE files; consistent with the notion that the file/hierarchy relationship is one-to-one. Since an IUE file will normally contain a flattened series of IUE Class Instance descriptions and references to those descriptions, the convention has been adapted that the final IUE Class Instance description or reference in a file will be considered the root of the hierarchy. The standard algorithm for traversing a network of IUE Class Instances will invariably attempt to make the root object the last one referenced in the file.

File Identification

An IUE File Identifier is constrained to appear at the very beginning of the file, with no spaces or newline characters embedded. This is so that a file sniffer may depend on the first 29 characters of the file being "(IUE-Exchange-Format-Version)". Case must be strictly adhered to in this particular instance, again, so that file sniffers may be as simple and fast as possible. It will probably be adequate for file sniffers to examine the first four characters ("(IUE") in most instances.

Comments

Comments are introduced by a semi-colon (;) character, as in Common Lisp, and run to the end of line character, again as in Common Lisp.

Sequence numbers and Class instance names

When an IUE file writer describes an IUE Class instance (using a 'make' clause), it is assigned a unique positive integer. Most writers will probably start with one and increment the number for each Class instance described, but as long as the integers are unique, and all integers referenced are defined somewhere in the file, there is no other requirement. For files generated by humans, Class instances may have names instead of numbers, in which case they are not assigned integers from the sequence. These names must be unique within the context of the file, and have no meaning outside of that context; an IUE file reader may discard them once a file has been read. In a human-generated IUE file, both class instance numbers and names may be used.

Using an object

The (use ...) clause is the standard method for referencing an IUE class instance. A use clause may refer to an integer sequence number (described in the previous paragraph), an IUE Class instance name, or an external object (using an external clause.) The object need not have been defined at the time that a use clause refers to that object; in such cases an IUE file reader will place the reference on the list of presently unresolved references, which are to be cleaned up by the time that the file has been completely processed. A standard algorithm for forward reference handling is provided in an Appendix.

Examples of (use ...) clauses:

```
(use 12)
(use "foo-bar edge")
```

External objects

An (external ...) clause may be used inside a (use ...) clause to include files. Any file included must be in the same directory as the file containing the (external ...) clause. The last Class instance listed in the external IUE file by either a (make ...) clause or a (use ...) clause will be the one referenced by the (use (external ...)) clause.

Example of an (external ...) clause:

```
(use 23 (external "my-cube.iue"))
```

In this example, the file my-cube.iue is processed and the last IUE Class instance made or used in the file is returned, and assigned sequence number 23. This sequence number may be used elsewhere in the file - but it must not appear in any (make ...) clause in the same file.

It is expected that IUE file readers will keep a list of files and the last objects referenced by them; this way, when an external reference is made, a check can be made to see if the file has already been read; otherwise multiple copies of objects might be created.

Making an object

A (make ...) clause consists of the reserved word 'make', followed by an identifier corresponding to a Class name from the IUE class hierarchy, followed by either a sequence number or a string, and finally the slots and attributes for this IUE object and their contents. Attributes are items not provided for in the IUE standard that an IUE user may wish to attach to IUE objects. Examples of (make ...) clauses and slots appear in the next section.

Slots

Slots may contain a variety of data types; these include simple types like bit, int, and float, more complex types such as string and vector, and may contain objects or lists of objects.

A slot clause consists of the word 'slot', followed by the name of the slot and by the content of the slot. If a slot refers to an IUE Class instance, then a use clause will be emitted referencing the object. If the object does not exist, then in a human-generated IUE file, a make clause may be inserted describing the object. A human writing an IUE file may recursively descend through the structure, writing make clauses as IUE Class instances are referenced for the first time. A program generating an IUE file is required to generate a flattened file description in which make clauses are never nested; this is done because extremely large, deeply nested files may otherwise impose unreasonable memory management demands on IUE file readers.

Example of a simple Make clause for a 3D Vertex located at [45.3, 23.8, 3.4] (the vector data type will be detailed later in this document):

```
(make 3D-vertex 52
  (slot p
    (vector 3 float 45.3 23.8 3.4)
  )
)
```

Attributes

Attributes are very similar to slots, but being user defined items, they are not described in the IUE document. It is the responsibility of the IUE user to insure that any attributes can be properly written and read.

Attributes also differ from slots in that since they are not described in the IUE document, their data type is not known until they are encountered during input; for this reason, the attribute type is included in an Attribute clause.

Example of a nested Make clause for a 3D edge with 2 previously undescribed vertices:

```
(make edge "my edge"
  (slot v1
    (make 3D-vertex 35
      (slot p (vector 3 float 3.45 -2.34 7.3298))))
  (slot v2
    (make 3D-vertex 36
      (slot p (vector 3 float 5.732 3.21 -2.3))))
  (attribute edge-name
    string "NE Edge of object Foo")
)
```

Example of the same description, flattened as required for machine input, and using the hard slots inferiors and superiors, as is more appropriate with machine generated IUE files:

```
(make 3D-vertex 2
  (slot p
    (vector 3 float 3.45 -2.34 7.3298))
  (slot superiors (list (use 1)))
)

(make 3D-vertex 3
  (slot p (vector 3 float 5.732 3.21 -2.3))
  (slot superiors (list (use 1)))
)

(make edge 1
  (slot inferiors
    (list (use 2) (use 3)))
  (attribute edge-name
    string "NE Edge of object Foo")
)
```

It is recommended that lower level objects be created immediately before higher level objects, in order to keep the list of unresolved references reasonably small.

Example of a 2D 1-chain² which contains 3 edges:

```
(make 2D-vertex 3
  (slot p (vector 2 float 1.0 2.3))
  (slot superiors (list (use 2))))
```

² A 1-chain is a sequence of connected line segments.

```

(make 2D-vertex 4
  (slot p (vector 2 float 2.0 2.3))
  (slot superiors (list (use 2) (use 5))))
(make 2D-vertex 6
  (slot p (vector 2 float 7.3 8.2))
  (slot superiors (list (use 7) (use 5))))
(make 2D-vertex 8
  (slot p (vector 2 float -2.0 9.3))
  (slot superiors (list (use 7))))

(make edge 2
  (slot inferiors (list (use 3) (use 4)))
  (slot superiors (list (use 1))))
(make edge 5
  (slot inferiors (list (use 4) (use 6)))
  (slot superiors (list (use 1))))
(make edge 7
  (slot inferiors (list (use 6) (use 8)))
  (slot superiors (list (use 1))))

(make 1-chain 1
  (slot inferiors (list
    (use edge 2) (use edge 5)
    (use edge 7)))
  )
)

```

Vector types

As may be inferred from the previous examples, the vector clause is used to describe homogeneous sequences of class instances; an array representation is presumed. A vector clause begins with the word vector. The second item in a vector clause is the number of elements in the vector; the third is the data type of the vector. Vector elements are constrained to be of the same type as specified by the vector type, or in the case of bit types, the elements must be integers.

IUE Exchange Format provides only 1D vectors and vectors of vectors; matrices are represented by vectors of similar vectors (which are generally similar in both length and data type.) It is possible to represent matrices of arbitrary size and dimensionality using this mechanism without extension to the grammar for the IUE Exchange Format.

Example of a vector representation of an 1024x1024x8bit array:

```

(vector 1024 vector
(vector 1024 bit8
10 11 10 10 14 14 15 ...
)
(vector 1024 bit8
11 11 10 11 14 15 16 ...
)
...
)

```

Note that decimal integers are being used to represent bit values; a special syntax for bit values is not particularly necessary.

Lists

A list clause consists of the word list, followed by a sequence of simple types, vectors, and IUE Class instances. Lists of simple types will always be homogeneous. Lists of objects are always constrained so that all objects share a common superclass. These restrictions are intended to ease C++ implementation.

Characters and Strings

A simple data type for single characters has been intentionally omitted; this is because the Lisp and the C/C++ worlds have decidedly different notions of what is appropriate syntax. A character string containing exactly one character is more than adequate for representation of a single character.

String is intentionally limited to printable ascii characters; by implication, strings may not presently contain end-of-line characters or tabs. Strings may not contain double quote characters. Strings are written between double quote (") characters.

Default Slot Values

Default slot values may be specified for classes and subclasses using the default-slot-value clause. These defaults will be used whenever the class is instantiated and a slot value is not explicitly provided; the defaults may be changed with a new default slot value form at the top level in an IUE file. To set a default neighborhood³ 3d-ordered-point-sets, one would use a form such as the following:

```

(make 3d-linesegment-neighborhood 23
  (slot span 3.3))
(default-slot-value
  3d-ordered-pointset
  3d-nghbrhood (use 23))

```

Floats

The syntax for floats is a subset of those of Common Lisp, C, and C++, thus permitting it to be parsed by the standard tools of any of those languages. It is expected that IUE floats will always be double precision floats.

Reserved Words

The number of reserved words has been kept to a minimum and the grammar designed so that changes to the IUE hierarchy will not necessarily force changes to the grammar for the exchange format; in particular, IUE class names and slot names are not reserved words.

User-defined IUE classes

A restricted form of class description is provided so that IUE users may describe their extensions to the IUE hierarchy. Such descriptions will be limited to class inheritance and slot definitions; no provision will be made for transmitting code fragments. An example follows:

³ A 3d-linesegment-neighborhood is a linesegment joining a pointset in a one-dimensional sequence. Points are not considered *connected* if the span distance is exceeded.

```

(class my-new-iue-class
  (inherits
    iue-class1
    iue-class2
    iue-class3)
  (slots
    foo float
    bar integer
    baz bit)
)

```

A lisp system can, of course, create such classes on the fly during IUE file input. A C++ system will have to take extra steps; a preprocessor will have to locate the class descriptions in the IUE file and emit a C++ header file fragment. The person(s) managing the IUE system at the destination site will be responsible for integrating this C++ code fragment with their system.

New IUE Classes must be defined before they are referenced.

Appendix 1

Output Algorithm

Using this algorithm to write an IUE class instance will produce a properly ordered flat file with the class instance for the IUE Class Instance selected appearing last in the file, which is proper organization for (external ...) clauses. Infinite loops due to circular references are avoided as objects who have sequence numbers assigned already have either been written or are on the stack waiting to be written, and thus do not need to be revisited. The algorithm is depth-first in character.

Data Structure required:

sequence number hash table – key is IUE-Class-Instance, datum is sequence number

method Output-object(IUE-Class-Instance, output-stream)

1: [check for previous visitation] if object already has sequence number stored in hash table then return

2: [assign sequence number] obtain sequence number and place in sequence number hash table

3: [for all slots] if slot contains object, list of objects, or vector of objects then for each sub-object recursively invoke Output-Object on sub-object

4: [for desired attributes] if attribute contains object, list of objects, or vector of objects then for each sub-object recursively invoke Output-Object on sub-object

5: [write instance header] "(make class-name sequence-number ..."

6: [for all slots write] "(slot slot-name slot-value)"

7: [for desired attributes write] "(attribute attribute-name attribute-type att-value)"

8: [write instance close] ")"

Input Algorithm

This reader will handle both flat and deep representations of data structures, correctly restoring circular references. If names are to be handled as well as sequence

numbers, then a C++ implementation will need to double up hash tables (this is not necessary in a Common Lisp/CLOS implementation.) Implementation will be somewhat different in a Lex/Yacc driven implementation, but details of the restoration of the circular references will be identical.

This algorithm presumes that an implementation can support 'empty shell' class instances, whose slots have not yet been filed in. If an implementation cannot support such IUE Class Instances, but can create an inferior object without knowledge of its superiors (e.g., create edges given vertices but not yet knowing 1-chains), then it will be necessary to provide an intermediate 'storage class' in which to stash Class information such as slot contents, until the object description has been read in allowing the IUE Class Instance to be created. Such a two stage process may necessitate doubled hash tables for storage of intermediate information, and cause some processing steps to be slightly delayed. Such a two stage process is used in the Yacc/Lex prototype reader for Geometer Jr.

There are many implementation details such as handling slots which contain lists of references (both resolved and unresolved) that are not handled; some creativity may be required of the implementor, although there are no insurmountable problems (just a few irritating ones.)

Data structures required:

sequence number hash table – key is sequence number, datum is empty-shell IUE Class Instance

unresolved reference hash table – key is sequence number of as yet undefined instance; datum is a list of references in the form (sequence number, slot-name)

call Function Read-Object for each (make ...) in the input stream:

Function Read-Object(input-stream) returns IUE-Class-Instance

R1: [make instance] create appropriate shell of a IUE Class instance based on IUE class type from (make ...) clause

R2: [record sequence number] put Class Instance shell and sequence number in sequence number hash table

R3: [for all slots and attributes] if make clause encountered then recursively invoke Read-Object on it, and set slot value to return value else if a sequence number is in the sequence number hash table then set slot value else put current sequence number, slot name, and sequence number of undefined object on the appropriate list in the unresolved reference table

R4: [check for references that can now be resolved] if sequence number for newly created IUE Class Instance appears in unresolved reference hash table, then fill the slots in the appropriate class instances and remove the associated triple from the table.

R5: [return] newly created IUE Class Instance as result

Appendix 2

IUE Exchange Format grammar:

```
<IUE-Exchange-Format-File> ::= <IUE-File-Identifier> { <top-object> }

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<letter> ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P
           | Q | R | S | T | U | V | W | X | Y | Z | a | b | c | d | e | f
           | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v
           | w | x | y | z

<standard-type> ::= vector | slot | float | integer
                 | bit32 | bit24 | bit16 | bit8 | bit
                 | type

<reserved-words> ::= make | use | list | slot | t | nil
                  | <standard-type>
                  | IUE-Exchange-Format-Version
                  | default-slot-value
                  | slots | inherits | external

<digit-sequence> ::= <digit> { <digit> }

<sign> ::= + | -

<float-exponent> ::= e <digit-sequence> | E <digit-sequence>

<dotted-digits> ::= <digit-sequence> . <digit-sequence>
                  | <digit-sequence> .
                  | . <digit-sequence>

<unsigned-float> ::= <dotted-digits>
                  | <digit-sequence> <float-exponent>
                  | <dotted-digits> <float-exponent>

<float> ::= <sign> <unsigned-float>
          | <unsigned-float>

<string> ::= <double-quote> <printable-ascii-characters> <double-quote>

<integer> ::= <digit-sequence> | <sign> <digit-sequence>

<label> ::= <integer> | <string>

<identifier> ::= <letter> { <identifier-char> }
               | <digit> { <hyphen-or-digit> } <letter> { <identifier-char> }

<identifier-or-type> ::= <identifier> | <standard-type>

<identifier-char> ::= <letter> | <digit> | -

<hyphen-or-digit> ::= - | <digit>

<IUE-File-Identifier> ::= ( IUE-Exchange-Format-Version
                           <digit-sequence> . <digit-sequence> )

<make-or-use> ::= <make> | <use> | nil
```



```

<obj-list> ::= { <make-or-use> }

<top-object> ::= <make-or-use> | <default-slot-value> | <class-definition>

<list> ::= ( list <list-tail> )

<list-tail> ::= <obj-list> | <int-list> | <float-list>
               | <vector-list> | <list-of-lists>

<int-list> ::= <integer> { <integer> }

<float-list> ::= <float> { <float> }

<string-list> ::= <string> { <string> }

<vector-list> ::= <vector> { <vector> }

<list-of-lists> ::= <list> { <list> }

<vector> ::= ( vector <element-count> <vector-tail> )

<vector-tail> ::= integer <int-list>
                 | float <float-list>
                 | string <string-list>
                 | vector <vector-list>
                 | list <list-of-lists>
                 | identifier <obj-list>

<element-count> ::= <int>

<class-definition> ::= ( class <class-name> <class-inheritance> <class-slots> )

<class-name> ::= <identifier>

<class-inheritance> ::= ( inherits { <class-name> } )

<class-slots> ::= ( slots { <slot-name> <identifier-or-type> } )

<make> ::= ( make <identifier> <label> <slots-and-attributes> )

<use> ::= ( use <label> )
         | ( use <external> )

<external> ::= ( external <string> )

<default-slot-value> ::= ( default-slot-value <identifier> <identifier>
                           <slot-value> )

<slots-and-attributes> ::= { <slot-or-attribute> }

<slot-or-attribute> ::= <slot> | <attribute>

<slot-descriptor> ::= ( slot <identifier> <slot-value> )

<slot-value> ::= <string> | <integer> | <float> | t | <list>
                | <vector> | <make-or-use> | <identifier-or-type>

<attribute-descriptor> ::= ( attribute <identifier> <identifier-or-type>
                              <slot-value> )

```

The Image Understanding Environment: Image Features

Keith Price and IUE Committee*
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273

Abstract

An IUE image feature is a spatial object that represents some information extracted from an image. These features include image points, curves, edges, lines, regions, and complex collections of these kinds of features, such as perceptual groups. Image features without reference to the underlying image correspond to basic objects in the spatial object mathematical hierarchy. The *image feature* class shares a common development with this mathematical structure. In practice, image features will be kept in collections (the features extracted from an image) and will be used as an element of that collection of through the use of a *spatial index*. This paper describes the philosophy behind the *image feature* with a few limited examples of how they are described and used.

1 Introduction

This paper assumes that the reader is familiar with the basic concepts used by the Image Understanding Environment (especially the basics of object oriented development and the basic classes used by the IUE) and should be read in conjunction with the other papers on the IUE included in these proceedings. The complete documentation on image features in the IUE requires many pages and this paper is intended to provide a general description of the image feature classes not to be the complete description.

Central to any Image Understanding research or application program is the extraction and use of image features. Users of the IUE include applications users who are primarily interested in the user interface, applications developers who will use and extend image features, and other developers who will implement the more basic programs. For each of these groups, the *image feature* class definitions will be important.

*The members of the IUE Committee are: Tom Binford-Stanford; Terry Boult-Columbia; Bob Haralick, V. Ramesh-U. Washington; Al Hanson, Chris Connolly-U. Mass.; Ross Beveridge-Colorado State; Charlie Kohl-AII; Daryl Lawton-Georgia Tech; Doug Morgan-ADS; Joe Mundy-GE; Keith Price-USC; Tom Strat-SRI.

Image features form a portion of the larger spatial object hierarchy and are directly related to the mathematical structure of the hierarchy. IUE users such as application developers will be interested in how to use and extend the image feature classes with little consideration of the larger spatial object hierarchy. The spatial object hierarchy and the relationship of image features to it developed through the series of meetings of the IUE Committee and forms a clean way to describe and implement the hierarchy. There are several reasons why developing image features within the spatial object hierarchy is crucial.

First, there is a natural correspondence between the sequence of topological constructs, e.g. *vertex*, *edge*, and *face* used for spatial objects, and the descriptions of image features for points and junctions, edges, and regions. Access to this topological representation is especially important for describing composite image features such as linked line segments, adjacencies between regions found in segmentations, and perceptual groups. Second, since image features generally correspond to the projection of three dimensional object models, it is useful to have the same underlying operations and representations used for both of them. Third, image features are characterized by having a wide range of possible attributes. We wanted to be sure that it would be possible for users to easily define and extend the attributes associated with image features. Many of the shape attributes associated with image features are described by fitting a curve or two dimensional shape corresponding to a spatial object.

2 Relationship to Spatial Objects

Image features are defined as spatial objects that represent some information extracted from an image. These features include simple features such as points, curves, and regions and complex collections of image features in perceptual groups. An image feature has very little meaning separate from its underlying image—without the relationship to the image, image features fit into the basic spatial object hierarchy. In practice, image features will be kept in collections (i.e. a collection containing all the edge features extracted from one image) and will be stored, referenced, and manipulated as elements of that collection. A common alternative for the collection would be the *spatial index* that allows image-like (efficient) access to the features based on position.

Since image features are usually kept in collections some of the slot values associated with image features are only stored once for the collection. For example, the *coordinate-system* associated with an individual *image feature* is that of the underlying image, and since most expensive processing with the image feature is fixed to the *coordinate-system* of the image, there is no cost associated with this information.

Methods for image features are inherited mostly from the *spatial-object* class. There will be varying needs for specializations for methods relating to extraction (from the image), property value computations (i.e. color), display on the image, spatial indexing operations, input, output, grouping, and the various iterators over sets of spatial objects (subsets, all in an area, etc.). Image features will also be used as the basis for the *region-of-interest* in the image processing operations.

Image features are developed following the mathematical structures used for *spatial-object* where the *vertex*, *0-chain*, *edge*, etc. of the *spatial-object* hierarchy have analogs in the image feature hierarchy. The most apparent difference between the mathematical structure of the *spatial-object* classes and image features is the inclusion of image related property values and the use of named object classes that correspond to the kinds of image features used in image understanding research and development programs. An important requirement of the IUE has always been to support rather than hinder research, so we do not define these objects in absolute final forms but indicate what property values (slots) are possible, the names that will be used, and the associated semantics. Between different programs, the slots may vary, but the creation, reading, and writing programs will allow for missing and extra information without harm. Our first description of image features will parallel the mathematical hierarchy of *spatial-objects*. Most image features fit clearly into this parallel hierarchy, but some may not be obvious at first glance.

2.1 Image Points as Special Cases

Image points are simple features for representing image positions. As was used for point in the spatial object hierarchy, the simple *image-point* does not inherit from the *image-feature* class, but is a primitive element that only contains the image positions. Sequences of points are stored in specialized versions of the *ordered-pointset*.

2.2 Vertex Objects and Image Features

The mathematical object *vertex* is a 0-D object. A 0-D image feature is a point in the image, possibly with some associated neighborhood. This point can be something extracted directly (e.g. points from an interest operator, a corner detector, etc.) or the point could be the result of an intersection of two lines (or *line-segments*). The primitive slots for an image feature vertex are the image positions, but, depending on the feature being modeled, many other property values are possible.

A common image feature corresponding to *vertex* is the *edgel*—extracted in a neighborhood and representing the step between two intensity surfaces. A simple edgel that encodes the yes/no result of an edge operator at

every pixel in the image is treated as a single point with no associated properties and is thus a *vertex*. The main aspect which distinguishes the class *edgel* from the basic *vertex* is that a local model for the geometry of the image intensity surface is assumed. The local neighborhood is defined by a disk about each potential edgel location.

In the IUE core, we allow a number of simple models for the intensity surface in the local neighborhood around each edgel location. The neighborhoods are characterized by the local structure of image intensity surfaces. The neighborhoods are defined as follows in order of frequency of occurrence in an image:

- A The interior of a single surface.
- B Two surfaces intersect at the edgel.
- C Three or more surfaces. Often corresponds to a corner.
- D The image intensity surface is too complex to be described by a simple model.

The intent of the cover with small disks is to divide and conquer, to make small neighborhoods that are sufficiently simple that it is feasible to describe the surface adequately over the neighborhood. This is a fine to coarse to fine approach. In this approach, the first level of disks is the smallest meaningful. It is simple to describe patches of class A that include a single continuous surface. Continuous patches are described in differential geometry by a tuple: (point, tangent plane, and curvature tensor).

It is reasonably simple to describe the compound surface over a disk of type B that includes two surfaces. This model is the most common representation used to define an edgel. Two surfaces are bounded either by a curve at which two surfaces intersect or by a limb, an apparent boundary. On a small disk, the boundary curve is locally straight. The model of type C is typically called a corner and is best represented as an attributed *vertex*.

In addition to the position slot that is inherited from *vertex*, the basic set of attributes that is associated with the edgel model are as follows:

Line Segment The parameters of the edgel line segment.

Tangent Vector For efficiency the local line segment may be represented as the tangent vector.

Tangent angle Another alternative is a quantized tangent angle.

Strength The local slope of the discontinuous transition between the two surfaces in the edgel neighborhood.

Left Surface Normal The surface normal of the intensity surface on the left of the edge boundary.

Right Surface Normal The right surface normal.

Covariance Matrix The variances of the parameters of the edgel model computed from the actual intensity distribution in the neighborhood. Gives a likelihood that the model holds for the local disk.

These point operation results can be grouped in a number of ways depending on the user's concept of the

underlying geometry of the boundary or region. The simplest group is the *0-chain* which is a set of vertices with an implicit linear ordering. Usually a full topological description is not applied until a local neighborhood analysis is done to "link" edgels into connected sets called *edgelchain*.

2.3 0-Chains of Image Features

The mathematical *0-chain* describes ordered collections of objects of class *vertex*, such as that formed by linking edgels into a discrete curve. For some purposes the *ordered-pointset* class may be used to group edgel sequences and for others the more complete *image-0-chain* is proper. The main difference is that the *image-0-chain* is a topological concept which supports algebraic operations on the points while the *ordered-pointset* is a set of feature points with no topological interpretations.

A sequence of corner objects also forms a *image-0-chain* but here there is often no local neighborhood relation assumed between the corners. However, the corners usually correspond to junctions of two or more edges and it is reasonable to use the topological structure of the *vertex* for the basis of the individual junctions.

2.4 Edge, One-Dimensional Image Features

The name edge has a clear meaning in describing graphs and in topology, and has a very different meaning in most image understanding work. In the IUE descriptions, we use edge for both of these meanings, but usually the context will indicate which one is meant. The mathematical *edge* corresponds to many image features, especially bounded line segments and curves. The *image-line-segment* and *image-curve-segment* are the most obvious objects in this class. These features may be computed directly from the image or derived from other image features.

The *image-line-segment* potentially has a large number of attributes, but the basic set of attributes include:

V0 and V1 The primary information of the line segment is the beginning and ending points. These are are vectors of image locations.

0-chain A 0-chain (ordered list) of the pixel locations corresponding to the line segment.

Strength The difference across the line segment, a float.

Segment-length The length across the line segment, a float.

tangent-angle The direction (in radians, a float) of the line segment.

2.5 1-Chain, Connected 1-D Features

The mathematical construction continues with the description of 1-Chains or ordered sets of 1-D objects such as sequences of line segments. 1-Chain structures can also intersect at junctions represented by objects of the class *vertex* and be extended to define closed region boundaries. In practice, it is often difficult to form complete topologies of these types in a bottom-up fashion.

An *image-0-chain* may be analyzed to produce a *image-1-chain* composed of *image-line-segments* that approximate the original curve and the individual segments maintain the order given by the original edge elements.

2.6 Face, 2-D Image Features

The *spatial-object face* corresponds to 2-D image features most clearly represented as a *2d-image-region* in an image. The low level representation of a region can vary according to the ultimate use due to time and space efficiency considerations, and can include point sets, binary masks, interval codes, boundaries, etc. Each of these can be derived from the others. A nice aspect of treating regions as faces is that all of the *edge* mathematical structures can be used directly to determine adjacent regions and represent the geometry of the image structures.

There are several typical attributes associated with *2d-image-regions*. Some of these are simple scalar and matrix attributes for describing shape such as Area (integer number of pixels), Euler number, Centroid (the 2-D point of the centroid), Scatter-Matrix-of-Pixel-Positions, and Compactness. Some shape attributes correspond directly to instances of 1D and 2D spatial objects which describe the shape of a region and are instantiated by applying their corresponding fitting methods to the edgel chain of a region. Other attributes such as average intensity, variance, and so forth, are computed using a spatial index to register a region with an image and to access the corresponding image values. These attributes are stored as a *gaussian-distribution* with two slots, mean and variance.

2.7 2-Chains, Linked 2-D Features

A *2-chain* is a sequentially linked *face* feature. It is likely that the more conventional region adjacency graph is the more effective data structure to group faces as image regions, although it is essential that the underlying mathematical descriptions of edges and faces be used. Composite regions produced by an image segmentation procedure are represented as multiply connected faces corresponding to a set of 1-Chains enclosing pixel areas in the image plane. Region merging operations are supported by the topological operations for removing common edges between faces. Merging operations require methods to access properties of adjacent regions sharing a common boundary. These descriptions of region adjacencies are similar to the attributes used with edgel models.

2.8 Blocks, 3-D Image Features

Images are typically 2-D objects, but with range sensors and time sequences, we can expect to deal with extending the *block* to image features.

2.9 Other Collections of Image Features

Not all collections of image features fit the definitions of the X-chains. For example, *perceptual-groups* formed by clustering some number of image features into perceptually meaningful structures may result in a *image-0-chain* or *image-1-chain* where points or lines are grouped into

single curves, but frequently they are grouped into either simple shapes with few features (e.g. two parallel lines, rectangles, etc.) or clustered into an area feature where order and linking are unimportant. The IUE will provide direct support for basic groups of image features through the *2d-segment-pair*, *2d-segment-triple*, and *2d-segment-quad* classes. The specific shapes will be handled by classes such as *2d-segment-parallel*, *2d-image-corner*, *2d-segment-junction*, *2d-u-shape*, etc. which will inherit from the appropriate *spatial-object* class and contain the links to the image features that contributed to the object.

Perceptual grouping produces hypotheses that include cluster of interior cells of a region and n-tuple of edgels along a smooth curve boundary between surfaces.

Basic images features may be grouped by a variety of properties such as proximity, alignment, curvature, etc. Many different data structures can be important to use in this grouping process, such as K-D Trees and quadrees. Additionally, the Hough transform performs directional grouping. Proximity and directional grouping also often use different forms of spatial indexing. One example of the grouping process involves the generation of two sets of hypotheses from a tuple of edgels. The first set is that there is not a smooth curve between two surfaces. The second set is that there is a smooth curve between two surfaces. Hypotheses for individual edgels are that they are on the curve, that they are random spatially-invariant as a result of camera noise, and that they are "clutter," non-random and not on the smooth curve, e.g. another edge.

3 An Example Image Feature

To illustrate the construction of a class in the image feature hierarchy we will use the *2d-image-line-segment*. This description shows the level of description available to the system developers and illustrates how slots and methods are inherited in the construction of objects. It must be pointed out that the format in the complete description documents is superior to what is given in this paper.

3.1 2d-image-line-segment

A major structure in image segmentation algorithms. Many of the slots are associated with the orientation of the line and there is a standard orientation ambiguity which arises in image coordinate frames. At times it is convenient to have the image coordinates with x along the increasing image column index and y downward along the increasing row index. This coordinate is left handed, and alters the meaning of the segment orientation. The sense of the coordinate frame is provided by its definition at the level of *spatial-object*. It is assumed that the inverted coordinate frame is normally used, i.e., 'y' downward. In order to put the features into a standard right-handed cartesian coordinate system for later processing, the 'y' orientations, θ_y , must be transformed to $\theta_y + 180$ which is a method on *coordinate-transform*.

- Superior Class *image-line-segment*

- Pseudo slots – the slots that are added with this definition.

- *tangent-angle-x float* Another alternate orientation specification for the line segment. The angle in radians of the line segment. The sense is counter-clockwise with respect to the x-axis.
- *tangent-angle-d-x float* Same as *tangent-angle-x* except the angle is in degrees.
- *tangent-angle-y float* Another alternate orientation specification for the line segment. The angle in radians of the line segment. The sense is counter-clockwise with respect to the y-axis.
- *tangent-angle-d-y float* Same as *tangent-angle-y* except the angle is in degrees.
- *slope float* The direction of the line represented as the slope in the image coordinate space.
- *x-intercept float* The position where the segment intersects the X axis, or the column value when the row is 0.
- *y-intercept float* The position where the segment intersects the Y axis, of the row value when the column is 0.
- *rho float* Hough Transform representation, distance.
- *theta-x float* Hough Transform representation, angle of the *2d-line-segment* normal in radians with respect to the x-axis.
- *theta-x-d float* Hough Transform representation, angle *theta-x* in degrees.

The inheritance structure shows how the slots for this object are derived. Most slots are derived from the objects earlier in the hierarchy with the direct 2-D image related slots added at this stage. Additionally, many of the slots are implemented as "pseudo" slots rather than as "hard" slots. That is, these behave like slots in terms of storing the values, but do not take any space if they are not needed. Additionally, some slots are defined early in the hierarchy and are refined as the hierarchy is developed (e.g. centroid changes from the general point to the more specialized *nd-image-point*). Slots such as *coordsys* (the coordinate system for the spatial object) are usually the same for all image features corresponding to one image (and are the same as the image).

<i>Inheritance Structure:</i>			
Class where Defined	Slot Name	Type of Slot	How Implemented
spatialobject	coordsys	pointer(coordinate-system)	hard
spatialobject	bounding-box	aligned-box-neighborhood	hard
spatialobject	centroid	point	hard
image-feature	centroid	nd-image-point	hard
image-feature	image	pointer(image)	hard
parametric-curve	domain	pointer(spatial-object)	hard
parametric-curve	range	implicit-curve	hard
parametric-curve	parametric-mapping	pointer(function)	hard
parametric-line	range	implicit-line	hard
parametric-line	lmat	vector[n](vector[2](float))	pseudo
2d-parametric-line	range	2d-implicit-line	hard
2d-parametric-line	lmat	vector[2](vector[2](float))	pseudo
topology-node	superiors	list(pointer(topology-node))	pseudo
topology-node	inferiors	list(pointer(topology-node))	pseudo
edge	0-chn	0-chain	pseudo
edge	1-chn	list(pointer(1-chain))	pseudo
edge	v0	pointer(vertex)	hard
edge	v1	pointer(vertex)	hard
image-line-segment	tangent-vector	vector[n](float)	pseudo
image-line-segment	fitting-tolerance	float	pseudo
image-line-segment	edgel-fit	float	pseudo
2d-image-line-segment	tangent-angle-x	float	pseudo
2d-image-line-segment	tangent-angle-d-x	float	pseudo
2d-image-line-segment	tangent-angle-y	float	pseudo
2d-image-line-segment	tangent-angle-d-y	float	pseudo
2d-image-line-segment	slope	float	pseudo
2d-image-line-segment	x-intercept	float	pseudo
2d-image-line-segment	y-intercept	float	pseudo
2d-image-line-segment	rho	float	pseudo
2d-image-line-segment	theta	float	pseudo
2d-image-line-segment	theta-d	float	pseudo

Spatial objects in the Image Understanding Environment

J.L. Mundy and IUE Committee*

Box 8

G.E. Corporate Research and Development
Schenectady, NY 12309

Abstract

A major insight achieved by the IUE committee is the concept of the "spatial object". Objects to represent typical IU features such as points, lines, arcs, volumes, etc., have common spatial attributes and operations. The concept of "spatial object" is an effort to abstract these properties into a compact set of generic classes. In this paper, we provide an outline of the design of spatial objects in the IUE.

1 Introduction

A major insight achieved by the IUE committee is the concept of the "spatial object". Objects to represent typical IU features such as points, curves, surfaces and volumes, have common spatial attributes and operations. The concept of "spatial object" is an effort to abstract these properties into a compact set of generic classes. Essentially, a spatial object is a point set in n -dimensions. The point set has an associated coordinate frame so that projections and transformations can be applied to the point set. It is not essential that the point set be a simple flat set, but can be represented as a hierarchical group or other relationship among groups of point sets. A polyhedral object represents a complex spatial object which represents a set of points in 3D space.

We divide the spatial object concept into major categories according to the dimension of the

point set and its embedding dimension. The major categories are, point, curve, surface and volume. These entities correspond to point sets with dimension, 0,1,2,3 respectively. The entities can be placed in coordinate spaces of their dimension or higher. For example, a plane can be placed in spaces of dimension 2 or higher. Composite structures such as polygonal curves or polyhedrons are represented by topological networks which contain pointers to the geometric elements. For example a 1-chain is a sequence of edges. An edge is a curve segment, bounded by two vertices. A vertex is a point with associated topological connections.

In addition to these basic structures, we also introduce the concept of neighborhood which is taken from the standard theory of point set topology. The various neighborhoods are needed to establish continuity and connectedness relations between points and other spatial objects. Any spatial object can act as a neighborhood, but we define a special hierarchy of typical neighborhoods with the idea that the rather simple intersection tests associated with neighborhoods will be implemented with hard coded method implementations and will short circuit most of the pointer chains associated with deep inheritance hierarchies.

In this paper, we provide an outline of the spatial objects in the IUE. The paper is organized in the following fashion. First we discuss the details of the class "spatial-object". In subsequent sections, we discuss implicit and parametric representations of spatial objects, representations for topological structures, and representations for solids.

*The members of the IUE Committee are: Tom Binford-Stanford; Terry Boult-Columbia; Bob Haralick, V. Ramesh-U. Washington; Al Hanson, Chris Connolly-Umass; Ross Beveridge-Colorado State; Charlie Kohl- AII; Daryl Lawton-Georgia Tech; Doug Morgan-ADS; Joe Mundy-GE; Keith Price-USC; Tom Strat-SRI; The committee effort has been funded by numerous DARPA grants and associated funding partners under contract to each institution.

2 Spatial Objects

2.1 Abstract Geometry

The theory of geometry is based on abstract concepts such as *point* and various point sets, e.g. *line*, *surface* ... etc. The theory of these structures can be presented and manipulated entirely in terms of formal predicate logic. For example, we can represent the axiom that there is always at least one point not on a given line¹

$$\forall \mathcal{L} \exists p \text{ Line}(\mathcal{L}) \wedge \text{Point}(p) \wedge \sim \text{In}(p, \mathcal{L})$$

However, the theory of geometry and topology is not very efficiently computed in such formal terms. For practical applications, it is necessary to provide a *model* for these abstract concepts which permits efficient computation of their properties and relations. A common model for geometry is the representation of a point as a tuple in \mathbb{R}^n . A line then becomes a set of points, where set membership is defined by,

$$\text{In}(p, \mathcal{L}) \equiv (p = \alpha p_a + \beta p_b) \quad (1)$$

i.e., two unique points determine a unique line by linear combination.

Now a particular concept can have many models. For example, a circle can be defined by $x^2 + y^2 - 1$ or $\rho - 1 = 0$ depending on the use of cartesian or polar coordinates in the model for the circle. The models are related by the fact that they represent the same abstract concept, the unit circle. For most applications, the most effective model for geometry and topology represents a point as an n -tuple from \mathbb{R}^n . This model is called the *n-Euclidean* model. The concepts such as curve and surface are sets of points or n -tuple sets.

In the development to follow, a *Spatial Object* will either be a point or a point set which can support a standard set of methods. Some of the more common basic methods of the spatial object are,

$\text{In}(p) [\mathbb{R}^n \rightarrow \{T, F\}]$ - A point p is in the set of points defined by the spatial object.

¹These is some confusion over the predicates, $\text{In}(p)$ and $\text{On}(p)$ in the mathematical literature. For example Hilbert uses "On" to mean that a point is an element of a line point set. On the other hand, common usage uses the predicate "In" to denote an element of a set. In the discussion here, we take the predicate symbols In and \in to be equivalent. We reserve the symbol On to refer to points which are *In* the boundary of a set.

$\text{Dimension}[\mathbb{R}^n \rightarrow Z]$ - The intrinsic dimension of the point set. For example a point has dimension 0, a curve dimension 1 and a surface dimension 2.

$\text{On}(p) [\mathbb{R}^n \rightarrow \{T, F\}]$ - For point sets which have a boundary, the method $\text{On}(p)$ is true for points which have a neighborhood containing at least one point, q , for which $\text{In}(q)$ is false. The concept of neighborhood is defined later.

$\text{Intersect}(\text{SO}) [\mathbb{R}^n \rightarrow \mathbb{R}^n]$ - Perform a Boolean intersection with the spatial object, SO.

$\text{Compose}(\text{SO}) [\mathbb{R}^n \rightarrow \mathbb{R}^m]$ - For parametrized spatial objects, it is meaningful to use the range of one spatial object, O_1 , as the domain of another, O_2 . The dimension of the *Domain* of O_1 is n and the range of O_2 is m . For example, a polygon may be used as a region of interest in an image.

$\text{Nearest-Distance}(p)$ - For point sets which have a metric defined², it is meaningful to compute the distance from a point to the nearest point, p , which is $\text{On}(p)$.

$\text{Transform}(\text{CS})$ - Transform all the points of the spatial object from the current coordinate system of the spatial object to the target coordinate system, CS.

Boundary - Returns the boundary of a spatial object. For example, the boundary of a curve segment is two vertices. Note that the boundary of a boundary is always empty or NIL.

$\text{Surface-Normal}(p)$ - When a point, p , satisfies $\text{On}(p)$ and the spatial object is a differentiable manifold then it is meaningful to compute the surface normal at a point. Many other surface differential properties can be defined at this level of abstraction.

These methods are used extensively throughout many IU algorithms and the IUE classes are designed to efficiently implement these basic methods. In most cases these efficient implementations will be based on sets of \mathbb{R}^n and associated algebraic operations.

²A metric is a function which defined on two points which obeys a set of axioms corresponding to the usual notions of distance. For example, $d(P_1, P_3) \leq d(P_1, P_2) + d(P_2, P_3)$.

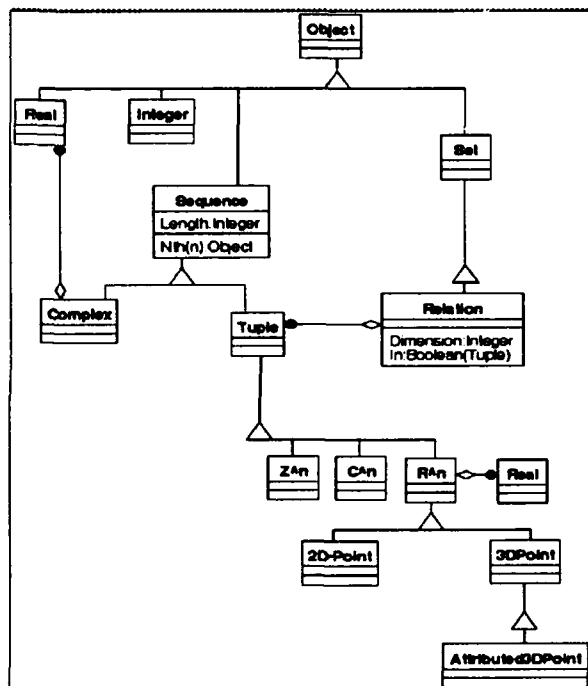


Figure 1: A portion of the IUE class hierarchy which defines the fundamental notion of relations and point sets.

2.2 The nDPoint

The *nDPoint* is the fundamental element of geometry. In the IUE the point is modeled as a tuple of real numbers, i.e. an element of \mathbb{R}^n . Each real number in the tuple corresponds to a coordinate value with respect to the coordinate-system associated with the spatial-object. A simple point can be considered to be a spatial-object with a default coordinate-system where each axis is just the real number line. A set of such points can be constructed and associated with a global coordinate-system where the axes have defined units and axis types, along with relationships to other coordinate systems. We will often use the mathematical structure of \mathbb{R}^n as a model for the theory of points, curves and surfaces. The hierarchy of Figure 1 illustrates the relationship of points and attributed points to the basic classes. In truth, there is little which distinguishes a *nDPoint* from the class *RealNTuple* and we may be able to dispense with the extra specialization layer. Arithmetic on elements of \mathbb{R}^n provides an algebraic representation for point set operations. In the example of Equation 1 shown earlier, a linear combination of two points, repre-

sented as vectors of \mathbb{R}^n , defines the points of a line. An *Attributed-nDPoint* is the same as point, except that a set of dynamic attributes are associated with the point. These attributes typically arise in the context of segmentation and correspond to properties such as texture, contrast, or differential geometry properties. For example an *Edge* is considered to be a point with attributes describing the local pixel neighborhood around a discontinuity.

2.3 Coordinate Systems

When we have a large number of simple objects, such as an *nDPoint*, it is not efficient to associate a coordinate system with each point individually. Instead, we group the points into a set and then associate the coordinate system with the set as a spatial object. In this case, we make use of the generic \mathbb{R}^n tuples to save the cost of linking each point to a coordinate system. Similarly, we do not attach a coordinate system to \mathbb{R} for many mathematical definitions. For example, it is not meaningful for π to have units. Sometimes it is warranted to associate a coordinate frame with a single point, as in a trajectory or track. We do not rule out the idea of a single point being a spatial object but to be an equivalent data type, it must be represented as a point set with one element.

The process of associating a coordinate system with a spatial object is achieved by associating attributes with each element of the \mathbb{R}^n tuple. Attributes are attached to define the dimension and type of each coordinate axis through the class *AxisType*. For example, a coordinate might be distance (dimension) measured in meters (units). Together these make up the attributes for a component of the \mathbb{R}^n tuple. This attributed \mathbb{R}^n is still a subclass of \mathbb{R}^n so it can be used as the range and domain sets of a spatial object. The relationship to the classes described in the section on coordinate systems is shown in Figure 2.

3 Implicit Point Set

The class *Relation* is specialized to form the concept of an *ImplicitPointSet* as shown in Figure 3. The tuples in the relation are instances of \mathbb{R}^n . The predicate defining the pointset will typically be a system of algebraic equations. For example the intersection of two ellipses in the plane generally defines up to four real points from \mathbb{R}^2 . When the

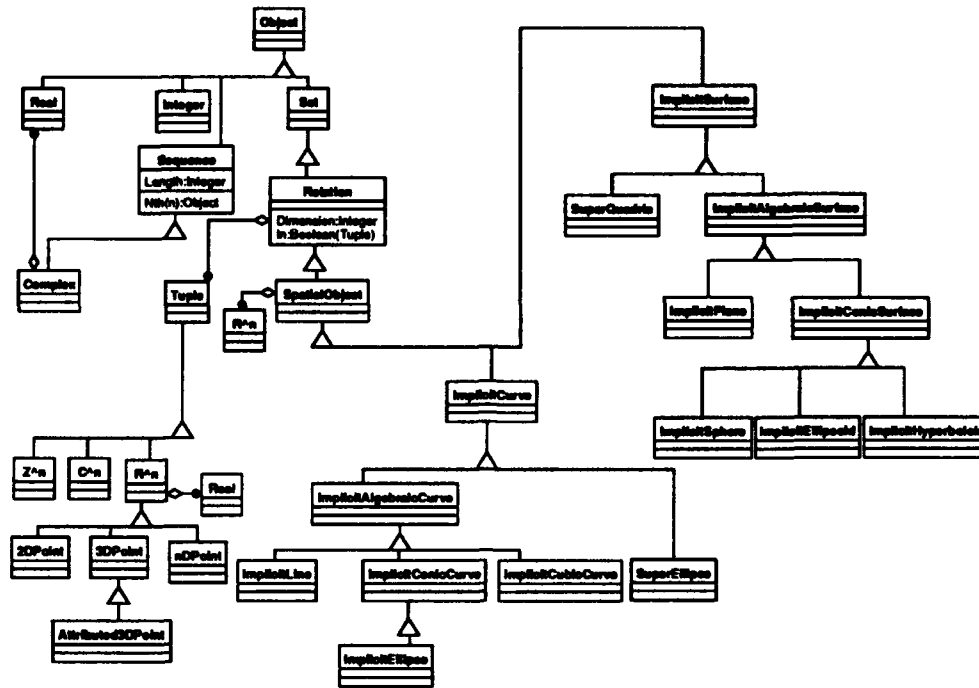


Figure 3: The implicit curve class hierarchy.

dimension and neighborhood properties of the implicit spatial object are preserved. These embedding mappings can be used to define a broad class of parametrised curves and surfaces. For example a conic can be defined as a parametrised curve in the projective plane as $[x_1, x_2, x_3] = [t^2, t, 1]$ which is the equation of a parabola. Then any conic can be generated by transforming this projective space with a homogeneous 3×3 matrix transformation. The 8 parameters of the matrix define a space of conic curves³.

5 Parametric Spatial Objects

A class of spatial objects can be defined in terms of the *function*. A parametric-entity in the IUE has the same interface as the type *Relation*. In this way, all of the standard class methods which apply to *Relation*, such as the predicates, *In* and *Intersect*, carry over to the parametric object classes. A function is a particular type of relation which defines a mapping between two sets of *n*-tuples, the *Domain* and the *Range*. The mapping associat-

ed with a function must satisfy the property that a given element from the domain maps to exactly one element of the range. A parametric entity uses a function, and the function is either a component of the parametric class or is intrinsic to the code implementing the class methods. A portion of the IUE class hierarchy illustrating the definition of the class *Function* is shown in Figure 4. In our use of the function⁴, as the basis for parametric spatial objects we further restrict the mapping to be order preserving and one to one. With these properties we can always find a unique point in the domain for a given point in the range and the natural dimension and neighborhood properties are preserved. This is a much stronger condition than is usually associated with the idea of parametric curves or surfaces. The curves here are perhaps more properly called "well-parametrised," where there is a unique inverse for each point in the range of the curve.

As an example, a parametrised line, \mathcal{L} , in the

³Actually, the space of conics is only five dimensional so this projective mapping is not uniquely invertible.

⁴Initially the IUE hierarchy was designed so that a parametric entity had the same class interface as a function. Subsequently, it was realised that a more uniform class interface is achieved by making parametric entities be a type of *Relation*.

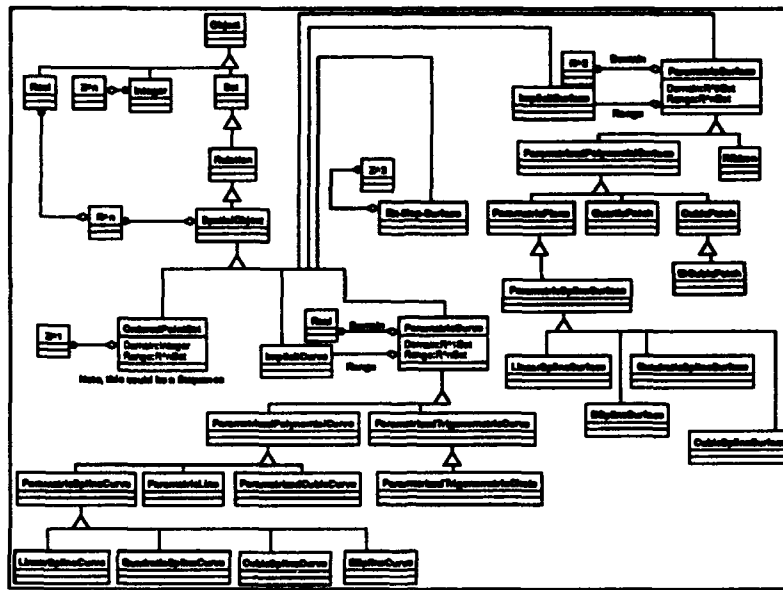


Figure 4: The class hierarchy for parametric curves and surfaces. These classes are based on functional mappings.

plane defines a mapping from \mathbb{R}^1 to \mathbb{R}^2 as follows,

$$\begin{aligned} x &= a_x t + b_x \\ y &= a_y t + b_y \end{aligned}$$

where $t \in \mathbb{R}$ is the domain and $(x, y) \in \mathbb{R}^2$ is the range. In general, parametrized curves are functions with a domain \mathbb{R} and represent mappings on to some nD point space. In this example of the line, the entire set of \mathbb{R} is used as the domain of \mathcal{L} . The range of the line in \mathbb{R}^2 can be further restricted by introducing a set of inequalities on the domain, e.g., $0 \leq t \leq 1$.

5.1 The method $\text{In}(p)$

The predicate $\text{In}(p)$ is decided by finding the value of the parameter, t , in the domain of \mathcal{L} and checking the inequalities. In the case of our example,

$$t = \frac{1}{2} \frac{(x + y) - (b_x + b_y)}{a_x + a_y}$$

So ultimately, the decision on the predicate $\text{In}(p)$ rests on an implicit point set which is defined by a set of equalities and inequalities on \mathbb{R}^n . Most often, these domains are simple balls or boxes in \mathbb{R}^n , as in our simple example with the line segment.

For curves defined by higher order polynomials, it is possible that a curve can cross itself, as in

the case of a parametric cubic curve. In order to make the parametrization uniquely invertible, we choose to introduce a vertex at the double point so that there is an invertible parametrization between endpoints.

This definition of $\text{In}(p)$ assumed that the point was on the line. This backward reference to the domain of the parametric line is not defined, if a point is not on the line. In general, the implicit form of a curve or surface is necessary to determine if a point is in the set when the set is embedded in a space with higher than the natural dimension. A natural approach is to use the implicit form of the curve as the domain of the parametric form. Then the method $\text{In}(p)$ for the range of the parametric form can be queried to determine if the point is on the curve before checking the bounds imposed by the domain inequalities. With our design for the parametric form of a spatial object, the *Domain* of an implicit form for a curve can be used as the *Range* of the parametric form since a *Range* slot or attribute is available for the parametric mapping function.

5.2 Composition

A function defines a composition operation where the range of one function is used as the domain

of another. In the usual notation, composition is represented as, $f(x) = g(h(x))$ where f is the composition of g and h . Here, the domain of f is the domain of h and the range of f is the range of g . We expect that the composition of functions will play a central role in the application of spatial objects in the IUE. For example, consider an image $I(x, y)$ which can be considered as a function from \mathbb{R}^2 onto \mathbb{R} . We also define a parametric curve, say a circle as a function from \mathbb{R} onto \mathbb{R}^2 . The composition of the two functions is, $I(\text{circle}_x(t), \text{circle}_y(t))$ and is a mapping from \mathbb{R} onto \mathbb{R} and represents the image intensities at the points of the circle which can be plotted as a one dimensional graph.

For the composition of a number of functions, the predicate $\text{In}(p)$ is computed by a chain of function inversions, until the decision is made on the point set representing the domain of the first function. Suppose we have a curve segment in the plane defined by, $C(f(g(h(t))))$ where $0 \leq t \leq 1$. To decide $\text{In}(p)$, we compute,

$$t = h^{-1}(g^{-1}(f^{-1}(C^{-1}(p))))$$

Then it is easy to check if $0 \leq t \leq 1$. Again, the ultimate decision of parametric point set membership rests on deciding the membership of an implicit point set, in this case, the domain of the function h .

5.3 Parametric curves

In the hierarchy of Figure 4 a number of types of parametric curves are illustrated. The classes are organized around the type of functions used to provide the mapping from the domain to range of the curve. The most common classes are defined in terms of polynomials of various degrees, for example a parametric cubic curve is shown in the figure. Another common class of projectively defined curves is constructed from rational polynomials. For example a parametric circle can be defined as the ratio of two second-order polynomials. Similarly, curves can be defined by trigonometric functions as in the case of a circle parametrized by polar coordinates.

We introduce the class `OrderedPointSet`, which is a sequence of points, to represent sampled curves or discrete pixel chains. Since the point set is ordered, it has all of the semantics of a parametric curve. That is, we can define a parameter along the curve which maps to points in the discrete point set.

Often, the point set is viewed as a set of samples from some continuous space and the class definition must provide the necessary structure to represent the neighborhood properties of the samples. For example, in the case of pixel chains, each point is associated with the square pixel neighborhood in the image. The computation of intersection or incidence is carried out by taking into account the intersection or incidence of pixel neighborhoods.

It is also common to define one-dimensional neighborhoods about each point by connecting each point with a straight line segment which is the minimal assumption about the continuity of the curve. Smoother assumptions can be introduced, such as higher order derivative continuity, e.g., C1 and C2, as well as global assumptions about the curve. For example, the samples are taken from a bandlimited curve and the original curve can be recovered exactly by *Sinc*(x) interpolation.

5.4 Spline Curves

A spline curve is a sequence of polynomial segments with continuity conditions at each segment breakpoint, or knot. The polynomial segments are delimited by the breakpoints along the curve. These breakpoints are represented by the class `OrderedPointSet`. Each polynomial segment is represented as a linear combination of basis functions. Depending on what basis functions are used and the order of continuity, we can categorize splines as being linear, quadratic, cubic, B-spline etc. A general-parametric-curve is one where the curve is specified by a linear combination of user specified basis functions. The different subclasses for parametric-splines include: 2d-linear-spline, 3d-linear-spline, 2d-quadratic-spline, 3d-quadratic-spline, 2d-cubic-spline, 3d-cubic-spline, and B-spline.

5.5 Parametric surfaces

The covering problem

Parametric surfaces are generated by functions with two arguments, i.e. functions with domain \mathbb{R}^2 . The range can be any tuple space but usually we think of continuous surfaces in \mathbb{R}^n . Again a wide range of surfaces can be represented by polynomial functions. However these mappings do not in general cover an entire surface. For example, a parametric mapping for the surface of a sphere in terms of spherical coordinates results in a non-invertible

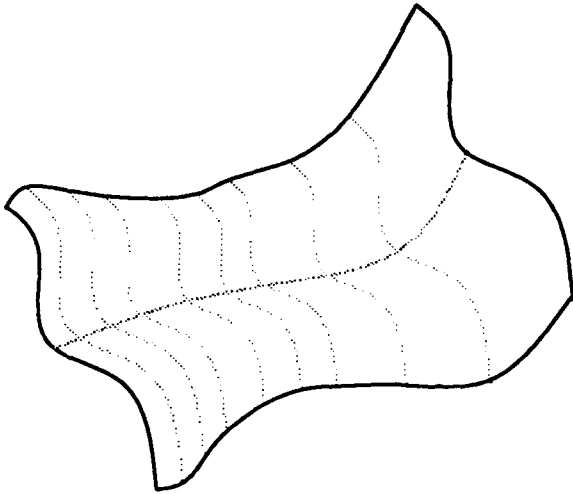


Figure 5: A parametrized surface patch generated by sweeping one curve segment along another. One approach to ribbon generation is to define a parametrized transformation.

mapping. However, two parametric surface patches with invertible mappings may be defined.

It is often necessary to use many parametric patches to make up a surface, since the particular representation is not able to accurately "fit" a desired shape, even though the topology provided by two or more patches is sufficient. For example, many industrial components with complex shapes are represented as composite surfaces made up of a network of BiCubicPatch instances. The patches are defined on a set of knots in a analogous manner to the spline curve case. The class OrderedPointSet introduced earlier is implicitly ordered as a sequence, since this ordering supports the bulk of applications for ordered point sets. Therefore it is necessary to add a new class, 2DArrayOrderedPointSet to maintain the knots for splined surface representations. A number of examples of spline surfaces are indicated in the hierarchy of Figure 4.

The Ribbon

The Ribbon is an unusual case since it is defined in terms of one curve segment which is swept along another curve segment by a parametrized transformation. An example is shown in Figure 5. The ribbon can be generated by introducing a coordinate system for both curve segments. Then the generator curve can be "swept" along the axis

curve by varying parameters of the transformation. The target coordinate system is a local coordinate frame along the axis curve. Usually the generator curve is maintained perpendicular to the axis curve at the intersection point. A free parameter of the transformation corresponds to the position along the axis curve. Taken together, these constraints define a parametrized transformation matrix which maps the generator curve at each point along the axis. The final result is a surface patch with two parameters, μ , position along the generator, and ν , position along the axis. As usual, the bounds on these parameters define the extent of the ribbon surface patch.

Discrete surfaces

In a similar manner to the discrete curve, a parametric surface can be defined by a bit map array where the points on the surface are generated by the two array indices. The boundary of the the surface can be defined by inequalities on the array index values. Alternatively, the elements of the array domain which have no corresponding surface point in the range can map to a range element which is reserved to indicate *undefined*.

As in the case of discrete curves, we often wish to consider this discrete representation as associated with a continuous space. To achieve a continuous representation it is necessary to maintain a neighborhood description for the sample points. A neighborhood definition is required to compute intersection or incidence methods. For example, for a discrete 3D surface, all points might have an implicit neighborhood defined by a sphere of given diameter. Then the method $ln(p)$ can be computed checking if p lies inside or on any of the spheres surrounding the sample points. Similarly to the discrete curve case, we can establish minimal two dimensional neighborhoods about each point by triangulating the mesh of points. Again, higher order continuous neighborhoods can be defined in analogy to the curve case.

6 Primitive Spatial Objects and Neighborhoods

In any discussion of continuous curves and surfaces as well as the sampling of these entities, it is necessary to introduce the concept of a *neighborhood*. In

the current representation a neighborhood is simply a primitive spatial object such as a line interval, a box, a disk or sphere. Since these structures are used so pervasively throughout the system, it is intended that these primitive structures can be included in any class with minimal cost in space and computation.

For example, in computing connected components on an image segmentation it is necessary to maintain a description of the square pixel neighborhood about each point. Since applications will involve millions of pixels for each image, it is essential that the representation of such neighborhoods be efficiently processed. On the other hand, it is important that the resulting topological structures are consistent and compatible throughout the system. It should be the case that a region boundary, extracted from an image, will have the same topological structure as a solid model and that image curves can be simply extruded to form solid surfaces.

In general it is desirable to have any spatial object act as a neighborhood, since in some applications the geometry of a neighborhood can be quite complex. For example the projected neighborhood of two conjugate pixels in a stereo image pair is a 3D trapezoidal prism in space. That is, any point within this volume will project to the corresponding pixel neighborhoods in the image. As another example, very complex composite neighborhood regions can be developed when the primitive neighborhoods are circular or spherical.

6.1 Topological Space

The fundamental concept of a *Topological Space* can be defined in terms of neighborhoods as follows:

A **Topological Space** is a set of points S along with the choice of a class of subsets \mathcal{N} of S , each of which is called a neighborhood of its points, such that,

- a) Every point of S is in some neighborhood.
- b) The intersection of any two neighborhoods of a point contains a neighborhood of that point.

The usual definition of topological spaces use open intervals, disks or spheres as neighborhoods as shown in Figure 6. It is easy to see that these

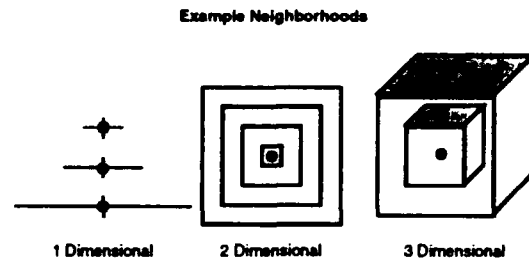


Figure 6: A topological space is defined in terms of neighborhoods. This figure illustrates a number of primitive neighborhoods for various dimensions.

neighborhoods satisfy the properties of a topological space. In the core IUE, we will implement neighborhood methods for line segments, boxes and rectangular prisms. These neighborhood geometries support efficient computation of boolean operations and predicates such as $\text{In}(p)$. The concept *near* is fundamental to define most of usual topological notions.

Let A be a subset of S and p a point of S . Then p is **near** A if every neighborhood of p contains a point of A .

The definitions, **Closed**, **Connected**, **Boundary**, can then be expressed in terms of *near*. For example, we can define A as **Closed** if A contains all of its near points. For more detail on these concepts and further definitions see Henle[4].

6.2 The Neighborhood Classes

The IUE designates a small number of basic neighborhood classes which are designed to be efficient. This portion of the hierarchy is shown in Figure 7. Neighborhoods are defined for various domain dimensions. Many of these neighborhoods are defined implicitly. A 0-Dimensional neighborhood corresponds to a point. The most pervasive 1-Dimensional neighborhood is the line interval, specified in terms of inequalities as

$$t_1 \leq t \leq t_2$$

Also provided is an oriented line segment to provide a local curve neighborhood as in the case of the Edgel. The neighborhood is specified as an oriented line segment by specifying a parametrized line and bound inequalities on the line.

The 2D analogy of the line interval is the rectangle which is aligned with the coordinate axes to

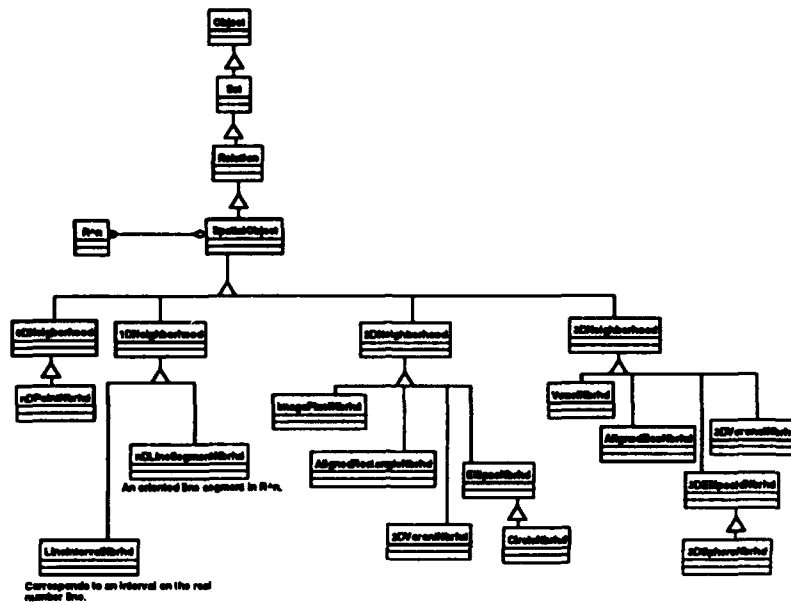


Figure 7: The IUE neighborhood classes. These classes are meant to be efficient for large volume applications and can be “mixed in” to other classes such as Image.

support efficient computation. A special neighborhood is defined for image pixel locations to support direct integer indexing of neighboring pixels. A more general 2D neighborhood is provided by the *VoronoiNbrhd* where the neighborhood region around a point is defined as the set of points closest to the point. These concepts carry over directly to three dimensional neighborhood domains.

It is assumed that the neighborhood is defined around each point in a spatial object as a result of the mixin. For a single point, the process amounts to constructing a “point-with-neighborhood” class. Often, the same neighborhood description, with the same parameters, is applied to each point in a point set. In the case of *Edge* neighborhoods, the orientation and possibly the length of the neighborhood varies from point to point. Similarly, a *Voronoi* neighborhood is, in general, a different region for each point in the set.

The neighborhood should in general be able to support all of the methods for a spatial object, such as Boolean intersection methods. The additional methods which each neighborhood class must support are summarized below. For the purpose of definition it is assumed that the neighborhood is defined about a point, *p*.

Near(*p*) - A predicate which is true if *p* is near the neighborhood defined by self.

Neighbors - Returns the adjacent neighborhoods of self. A set of neighborhoods is returned consisting of the neighborhoods connected to self. For example, the *Neighbors* of a pixel neighborhood, $n_{i,j}$, in a 4-connected image array are the pixel neighborhoods, $\{n_{(i-1),j}, n_{(i+1),j}, n_{i,(j-1)}, n_{i,(j+1)}\}$.

In general, there are adjacent point sets of various dimensionality. For example, the point of intersection of a line with a plane has a 1D set of neighbors on the line and a 2D set of neighbors on the plane. This situation is handled below in a more structured topological representation below⁵. At this basic level, it is assumed that the topological space is everywhere the same dimension.

6.3 Why Separate Neighborhood Classes?

In the discussion of neighborhoods, there is no real distinction between the neighborhood classes and the general concept of a spatial object. Indeed, we

⁵In the treatment of Section 8 the *neighbors* sets is divided into a number of called *superiors* and *inferiors*. The *superiors* are one higher dimension than the given entity and the *inferiors* are one dimension lower. Also there can be more than one adjacent pointset, e.g. the edges incident at a vertex.

intend that any spatial object can act as a neighborhood. On the other hand, we expect that the efficiency demanded of a neighborhood will require a special implementation and therefore we define these separate classes. Conversely, it is expected that the neighborhood classes are quite satisfactorily used as spatial objects and it can be considered that the neighborhoods are specializations of the appropriate spatial object class.

For example, a rectangular neighborhood is the set of points interior to a rectangle which is a specialization of four-sided polygons. However, we do not need to maintain the extra baggage associated with the bounding 1-Cycle and associated edges and vertices in order to perform neighborhood methods. Admittedly, we might choose to implement the rectangular planar face as efficiently as just assumed for the rectangular neighborhood and then the two concepts would be identical. In any case, the same number of specialized classes must be implemented.

7 Hierarchical and Sequence Groups

In order to proceed with our development of spatial objects, it is necessary to introduce several classes which define groups of the geometric primitives we have just discussed. The first group is the Sequence which is an ordered set of objects. A general approach in the IUE for implementing the access to groups is through the mechanism of inheritance. That is, an object becomes part of a group by inheriting the mechanisms of a generic node class defined for the group. In this case, an object becomes part of a sequence by inheriting the class `SequenceNode` as shown in Figure 8. The basic methods for `SequenceNode` are `Next` and `Previous`. The intention is that the node classes are designed to be very efficient at computing the direct access methods required by a node. More global queries such as the total number of items in the sequence should be referred to the Sequence class itself.

The second type of group we need to define is the `HierarchicalGroup`. The concept is a hierarchy of superior and inferior elements which form a tree. An typical example of a hierarchical group is the description of part-whole relations. For example if the human body is considered the root of the group tree, the inferiors are parts like head, trunk, arms,

legs, etc. The inferiors of head are eyes, nose, etc. The superior of leg is body. The case of multiple superiors occurs when a part is shared. For example the wrist may be considered as part of both the hand and arm.

As shown in Figure 8 an example global method of `HierarchicalGroup` is `Superior-p(Node, Node)`, which determines if one node is indirectly superior to another in the hierarchy by tracing up the tree. We define the class, `HierarchicalGroupNode` which efficiently implements access to the list of direct superiors and direct inferiors of a node. We will make immediate use of `HierarchicalGroup` in defining the topology of geometric structures. Also note that the usual notions of part composition is implemented naturally by a subclass of `HierarchicalGroup`.

The approach of inheriting a `HierarchicalGroupNode` to acquire the methods of `Inferiors` and `Superiors` is limited to one grouping for each entity. It does not make sense to inherit more than one hierarchical node. This creates a problem for situations where an entity is a member of a number of hierarchical groupings. The solution proposed so far is to have a multi-group node class for those structures which require membership in more than one hierarchy. A single hierarchy node class is also available for efficiency reasons.

8 Topology Groups

So far we have introduced the concepts involved in defining the primitive classes of points, curves and surfaces. The next step is to define the topological structures which are needed to construct composite curves and surfaces. The fundamental notion of topology is that of "connection" or adjacency. A simple example is provided by two line segments which share a common endpoint. The common point of incidence is viewed as a connection between the two line segments. In order to develop a consistent topology, the connection between two line segments is restricted to endpoints. That is, if two line segments intersect at all, they intersect at an endpoint.

8.1 The Vertex

These considerations lead to the definition of the class `Vertex`. The vertex is both a point in space as well as a connection. In order to represent these

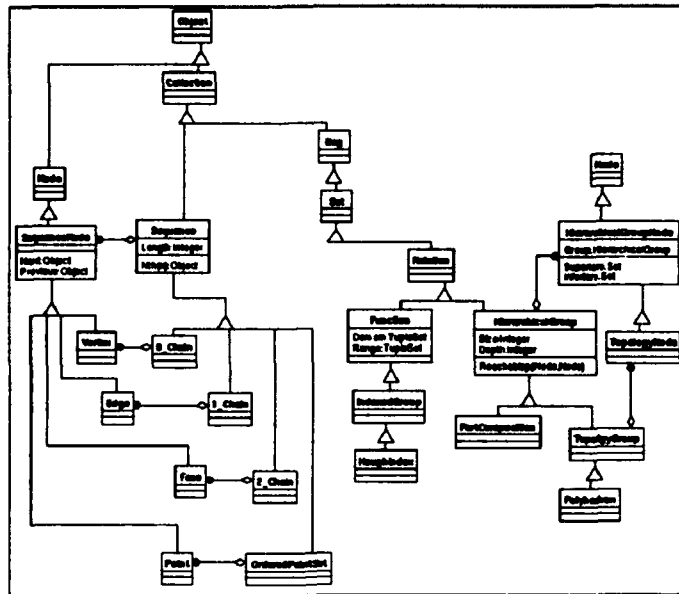


Figure 8: The definition of sequence and hierarchical groups.

properties, the vertex class is constructed by multiple inheritance from the class **Point** and **TopologyNode** as shown in Figure 9. The class **TopologyNode** is a special case of **HierarchicalGroupNode** which defines a set of **Inferiors** and **Superiors**. In our case, the vertex has no inferiors but has a set of superiors which are instances of the class **Edge**.

Each of these topology elements is illustrated in Figure 9.

8.2 The 0Chain

The **0Chain** is, strictly speaking, an unordered set of vertices. In most applications it is useful to establish an order on this set and thus we consider the **0Chain** to be a **Sequence**. The class **Vertex** therefore must inherit from the class **SequenceNode**. The concept of an ordered set of vertices is useful for various topology constructions. For example, when constructing a polygon from the set of its vertices, it is necessary to maintain a strict ordering so that the boundary of the polygon is consistently defined. The ordering is also necessary to maintain a consistent surface normal orientation.

The **0Chain** is also useful in the case of a curved edge, such as a spline, which may have a number of sample points on its interior between the endpoints. These interior points are not vertices, but they may be used to define a sequence of line

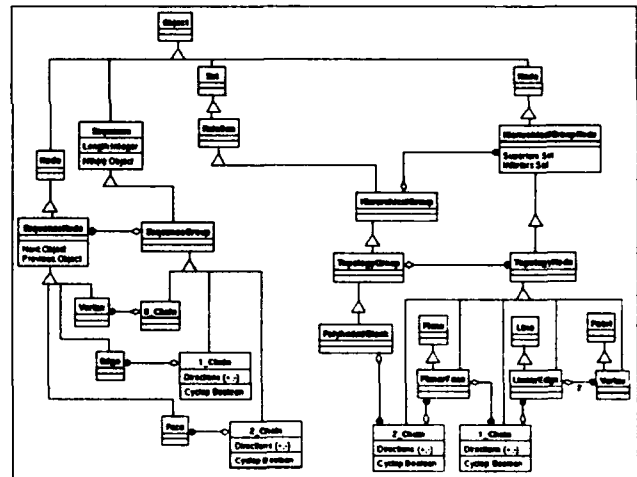


Figure 9: The various elements of object topology. Note that if a **1Chain** or **2Chain** is closed it forms a **1Cycle** or **2Cycle**. It is not necessary to introduce a new class but just associate the attribute "closed" with the chain structures. The structures are ordered top-to-bottom, left-to-right in the order of the inferior \rightarrow hierarchy.

segments(polygonal line) for display or intersection calculations. For these applications, this set of interior and boundary points are conveniently represented by a 0Chain.

8.3 The Edge

The class **Edge** is a bounded segment of a curve, where the boundary of the segment is defined by two vertices. These bounding vertices are established as the inferiors of the edge. The superiors of an edge are a set of 1Chains, where a 1Chain is a sequence of edges. The edge is constructed by multiple inheritance from a curve class as well as the classes **SequenceNode** and **TopologyNode** in order to support the necessary relationships. Note that an edge can participate in a number of 1Chains, as in the case of two polygons which are joined at a common edge.

The generic predicate $\text{In}(p)$ for the edge can be computed by parametrizing the curve in association with the bounding vertices. A typical choice is to define the parameter at the first vertex as $t = 0$ and $t = 1$ at the second. The predicate $\text{On}(p)$ is true when the point p is either of the endpoints since these points define the boundary of the edge.

When representing higher order curves it is possible for an edge to intersect itself somewhere other than a vertex. This event violates the requirements for a consistent topology and a vertex must be introduced at the point of intersection. The numerical computation of such events is difficult and errors can result in an inconsistent topology. Issues of this sort have impeded progress in CSG modeling for curved surfaces.

8.4 The 1Chain

The class **1Chain** has many applications within the IUE. It is the basic structure for defining composite curves, e.g. a polygonal chain of edges. The 1Chain is a sequence of edges with additional direction information associated with each edge. The direction values, \pm define the sense of traversal of an edge along the curve. If the traversal is from Vertex_0 to Vertex_1 of the edge then the direction is $+$, otherwise $-$. The 1Chain is also used to represent closed boundaries of surface regions or faces. A closed 1Chain, or 1Cycle, has one superior, i.e., the face which it bounds. The inferiors of a 1Chain are the edges in the chain.

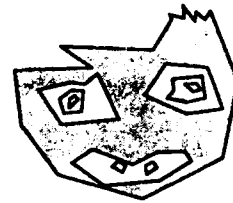


Figure 10: A bounded surface or face. An example is shown of a multiply connected face with interior holes.

Many of the topological properties required to define image features are provided by vertices, edges and 1Chains. The usual process of joining and recursively growing sequences of line segments can be handled by manipulating the 1Chain structure. Also, boundary corner events such as "T" and "Y" junctions are defined where image edge chains meet at a vertex.

Note that it may be necessary to maintain a number of topological descriptions for a pixel chain. The first level is the original sequence of pixel locations. Next, we may fit straight lines or splines to the pointset using a number of levels of fitting tolerance. Each fit, will produce a different 1Chain which should be associated with the other 1Chains as well as the original discrete set. This composite structure is analogous to a pyramid, since a larger error tolerance leads to fewer curve segments. As yet, we don't have an adequate understanding of the semantics of this type of grouping.

8.5 The Face

The class **Face** is a bounded surface region. The boundary may be multiply connected as shown in Figure 10. The face multiply inherits from classes **SequenceNode** and **TopologyNode**. The face inferiors are the 1Chains defining the face boundary. This definition of the face allows multiply connected regions with one outside 1Chain and any number of interior 1Chains as shown in Figure 10. The superior of the face is the 2Chain which is defined in the next section. One important application of the face structure is to define the boundaries of a region segmentation. In this case the surface is a discrete point set as well as the edge curves forming the boundaries. In the current definition, there is only one level of interior 1Chains. However, the case of "islands" within the holes of a face is sometimes encountered in image region segmentation.

This structure can be accommodated by arranging the 1Chains in a tree structure. Each level of the tree accounts for another layer of "inside" containment. However, it should be noted that this layered structure is not consistent with the usual notions of connectivity. In the case of image segmentations, the concept of "figure" and "ground" regions arise, where the ground region is taken to be a continuous surface. The figure regions occlude the ground region, but the ground is taken to be one connected surface even though only isolated islands may be visible. With the containment tree structure for the inner face boundaries, it is straightforward to produce the set of equivalent standard topology faces.

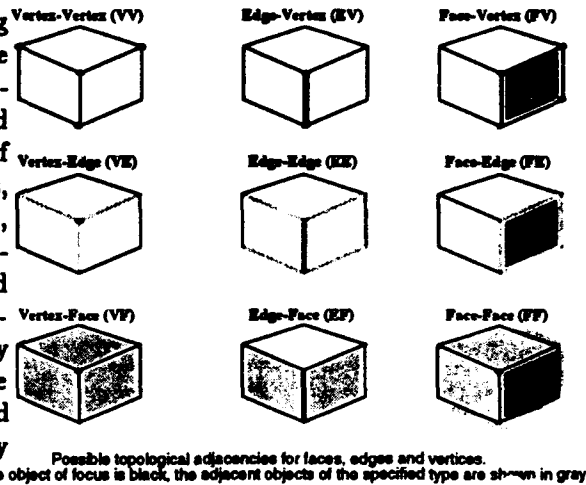


Figure 11: The nine possible topological adjacency schemes. The combinations result from a primary structure selected from { Face, Edge, Vertex } and an adjacent structure selected from the same set.

8.6 The 2Chain

In general, the face is used as the primitive element in any composite surface. As we pointed out earlier, it is always necessary to form composite arrays of surface patches in order to cover a closed boundary of space. The essential structure needed to define a composite surface is the 2Chain. The 2Chain is a sequence of faces which where adjacent faces in the sequence meet at a common edge⁶. If the 2Chain is closed, it is called a 2Cycle and defines a volume of space.

8.7 The Block

If a volume of space has interior cavities, then a number of 2Chains are used to represent the region boundary in analogy to the multiply connected face region. The class Block is introduced to contain the required 2Chain pointers as inferiors. The block is the highest topological structure and has no superiors⁷. Composite objects, such as articulated structures or different buildings in a site model, are usually constructed from separate blocks with each block embedded in a transform network. That is, each block has its own local coordinate system and a set of transforms to the other blocks or to a central coordinate system.

⁶Note that a single sequence cannot represent all adjacencies on a surface. However if two faces are next to each other in the sequence, they share a common edge.

⁷Note that we could naturally extend the representation to include 4 dimensional structures by joining blocks at faces to form 3Chains.

9 Relation to Other Topological Schemes

The IUE topology can be compared with other topological structures using the concepts derived by Kevin Weiler[5]. Weiler classifies various topological schemes in terms of the types of adjacency relations supported by the structure. There are three basic structures, vertex, edge and face. Consequently, there are nine combinations of primary and adjacent structural schemes. These combinations are illustrated in Figure 11. In the case of curved surfaces, it is impossible to recover the correct topological description of an object from the indicated adjacency information and it is usually necessary to provide ancillary structures to complete the sufficiency of the topological structure. The IUE structure provides a number of these adjacency relationships at the same time and provides a complete topological specification.

The IUE structure is also reasonably efficient in terms of storage. For example, the IUE structure takes $(4 + (7/2)m)N_f$ pointers where m is the number of edges per face and N_f is the total number of faces⁸. By contrast, a winged edge structure⁹

⁸A similar design to the IUE structure[3] represents the two endpoints of an edge as a 0Chain, but this approach considerably increases the pointer count.

⁹The winged-edge structure has an edge as the primary structure and edges as the adjacent structures.

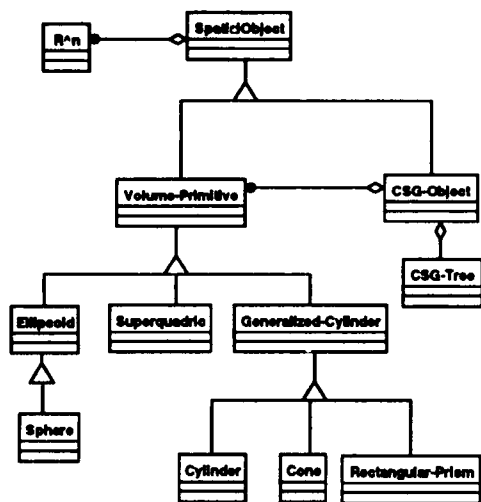


Figure 12: A partial class hierarchy for constructive solid geometry, CSG. Only a small sampling of possible solid primitives are indicated.

uses $4mN_f$ pointers. As the complexity of the face increases, as would occur in image regions, the IUE structure takes less space than winged-edge. The cross over is at about 10 edges per face. The IUE topology is similar to the usual notions of vertex-edge-loop-face-shell-block[1, 2]. However, the use of n Chains introduces the machinery of chain algebra which considerably clarifies the removal of "bridges" in topological configurations. The rigor introduced by the axioms of chain algebra assists the construction of algorithms for boolean operations.

10 Constructive Solid Geometry

The other major solid representation is in terms of solid primitives which can be considered blocks from the boundary point of view. These primitives are combined by attachment and boolean intersection operations to form composite objects. The resulting composite object is described by a tree where the nodes of the tree represent various primitives and partial constructions and the arcs of the tree represent boolean or attachment operations. This description is referred to as the Constructive Solid Geometry, or CSG, representation. A partial volume representation hierarchy is shown in figure 12. For example, the superquadric is a flex-

ible surface primitive whose shape is controlled by providing variable exponent values on the quadric terms.

The generalized cylinder is another major representational approach where the primitives are defined by a axis which can be a general space curve and a sweeping rule which defines the variation of the object cross section along the axis. For example, a cone is a generalized cylinder with a circular cross section and a linear sweeping rule along a straight line axis. The complete description of the object is maintained in a tree structure, called the CSG tree, where the leaf nodes are primitives and the root is the final object. Boolean operations are carried out between primitives at each interior node of the tree. This representation requires that Boolean operations are defined for each primitive which allows quite complex objects to be defined by a short description. For example a cylinder with a hole can be defined as the subtraction of one cylinder from another. Usually each primitive is associated with a bounding box (rectangular prism) which facilitates efficient checking for the possibility of intersection. The Boolean operations required for a CSG definition are, Union, Intersection, and Difference. Union is used to join two spatial objects into a single spatial object with no boundary at the join. Intersect(SO) is used to find the common point sets between two objects. For example the intersection of a cube and a cylinder is a (possibly) shorter cylinder. Difference(SO) - Forms the intersection of an object with \bar{SO} , the complement of the second object. In this case the difference of a cube and a cylinder is a cube with a hole.

References

- [1] C.M. Hoffman, *Geometric & Solid Modeling*, Morgan Kaufman, 1989.
- [2] H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison Wesley, 1990
- [3] C.I. Connolly, *GeoMeter 9.0: Reference Manual*, Sep. 6, 1991
- [4] M. Henle, *A Combinatorial Introduction to Topology*, W.H. Freeman and Company, 1979.
- [5] K. Weiler, "Topological Structures for Geometric Modeling," Ph.D. Thesis Rensselaer Polytechnic Institute, Troy, New York, August 1986.

Section VI

Workshop Reports

Computational Sensors:

A Report from the DARPA Workshop

Takeo Kanade and Ruzena Bajcsy*

Carnegie Mellon University and * University of Pennsylvania

1 Introduction

Computational Sensors combine computation and signal acquisition to improve performance and provide new capabilities that were not previously possible.

They may attach analog or digital VLSI processing circuits to each sensing element, exploit unique optical design or geometrical arrangement of elements, or use the physics of the underlying material for computation. Typically, a computational sensor implements a distributed computing model of the sensory data, including the case where the data are sensed or preprocessed elsewhere.

Recognizing the importance and potential of computational sensors, Oscar Firschein, DARPA SISTO, requested us to organize a workshop to bring together developers and users of computational sensors. The workshop was to define the state of the art, discuss the issues, and identify promising approaches and applications for this new technology. The workshop was held at The University of Pennsylvania on May 11-12, 1992. Approximately 40 people attended from academia, government, and industry. The workshop hosted several key presentations and followed them with group discussion and summary sessions. This workshop report presents a summary of the state of the art in computational sensors and recommendations for future research programs.

In Section 2 we discuss opportunities for computational sensors. Some computational sensor examples are reviewed in Section 3. Technologies, issues, and limitations are considered in Section 4. Section 5 discusses algorithms for computational sensors. Recommendations for future programs are given in the concluding section. The appendix includes a

bibliography of computational sensing created with input from the workshop participants.

2 Opportunities

Traditionally, sensory information processing proceeds in three steps: transducing (detection), read-out (or digitization), and processing (interpretation). Micro-electronics technologies will spawn a new generation of sensors which combine transducing and processing on a single chip - a computational sensor.

In machine vision, the basic approach has been to use a TV camera for sensing, to digitize the image data into a frame buffer and then to process the data with a digital computer. Apart from being expensive, large, heavy, and power-hungry, this sense-digitize-and-then-process paradigm has fundamental performance disadvantages. A high bandwidth is required to transfer data from the sensor to the processor. The parallel nature of operands captured in a 2D image plane is not exploited. Also, high latencies caused by this method, due to image transfer times, limit the usefulness of this method for high-speed, real-time applications. Combining processing on silicon wafers together with detectors will eliminate these limitations, and have the potential to produce a visual sensor of low-cost, and low-power with high-throughput and low latency.

The potential for integrating the transducing and processing of signals has been recognized for some time, but in the past, research and development in this area was driven mostly by curiosity or special use. Today, however, the advancement of VLSI and related technologies provides opportunities for us to harness this potential in new, broad, practical applications in image understanding, robotics,

and human-computer interfaces. Most importantly, VLSI technologies have become available and accessible to the sensor application community where we have recently observed a growing body of research in computational sensors.

Several computational sensors have been fabricated and demonstrated to perform effectively. Analog vision chips have been demonstrated which can detect a motion field, or continuously compute the size and orientation of an object. Three dimensional range sensing has been performed at a rate of 1000 frames per second using a chip containing an array of cells each capable of detecting and calculating the timing of an intensity profile. Sensor chips that mimic the human's fovea and peripheral vision have been fabricated and used for pattern recognition. Tiny lenses can be etched on silicon to focus light efficiently on a photosensitive area, or even to perform a geometrical transformation of images. Resistive networks and associated circuits on a chip can solve optimization problems for shape interpolation.

Computational sensors are not limited to vision use, but have applications in mechanical, chemical, medical and other sensors. Development of micro-mechanical pressure sensors and accelerometers has been underway for some time. An air-bag sensor for automobiles could become one of the first successful, mass-produced, low-cost computational sensors. It contains a miniature accelerometer and processing circuits in a chip. Processing could also be combined with micro-chemical sensors to detect water contamination, air pollution, and smells, while micro-medical sensors could measure blood chemistry, flow, and pressure.

Potential applications/markets of computational sensors are abundant:

- robot perception
- industrial inspection
- navigation and automobile
- space
- sensor based appliances
- medicine (e.g. patient monitoring)
- security and surveillance
- entertainment and media

- toy

Development of a computational sensor does not simply mean combining sensing capability with processing algorithms. It requires new thinking. Most of the current vision algorithms, for example, are strongly influenced by the fact that image data is provided in a stream and processed by instructions. Also, the concept of frame rate (ie., considering a certain number of discrete frames per second) is dominant in dealing with time varying events. However, a computational sensor can take advantage of the inherent, two-dimensional nature of the sensory data arrangement, the continuous time-domain signal, and the physics of the media (eg. silicon) itself for processing. This type of new thinking often results in a completely different, more efficient, orders-of-magnitude faster "algorithm". Many of the successful examples mentioned above and in section 3 are the results of such new algorithms.

Finally, computational sensors can create a fundamental change in the approach to the sensor system as a whole. When a sensor is bulky, expensive and slow, it is not affordable, both economically and technically, to place many of them within a system. The sensor system is forced to be centralized. If computational sensors can provide cheaper, smaller, and faster sensing units, we can place a large number of sensors throughout a system, such as covering the whole surface of a submersible vehicle. A new opportunity exists to make sensor systems more distributed, reliable, and responsive.

3 Computational Sensors: Some Examples

This section reviews computational sensor architectures that have emerged in recent years:

1. The focal plane computational sensor: Processing is done on a focal plane, i.e. the sensing and processing element are tightly coupled;
2. The spatio-geometrical computational sensor: Computation takes place via the inherent geometrical structure and/or optical properties of the sensor;
3. The VLSI computational module: Sensor and processing element are not tightly coupled, but processing is done on a tightly coupled module.

Many existing systems would fall into several of the above categories. Representative examples of each category are presented here.

Although most examples we give are of visual information processing, these considerations and techniques extend directly to measurement over the whole spectrum of electromagnetic radiation. In general, any other "imaging sensors" such as mechanical (e.g. tactile) or magnetic sensors, could also benefit from lessons learned when considering and designing computational sensors for vision applications.

3.1 The focal plane architecture

The focal plane architecture tightly couples processing and sensing hardware—each sensing site has a dedicated processing element. The sensor and the processing element (PE) are located in close physical proximity, thus reducing data transfer time to PE's. Each PE operates on the signal of its sensor. However, depending on the algorithm, each PE may need the signals of neighboring sensors or PE's. This concept corresponds to the SIMD paradigm of parallel computer architectures. In computational sensors, the operands are readily distributed over an array of PE's as they are being sensed.

Cell Parallelism

Gruss and Kanade [26] [27] [40] at Carnegie Mellon have developed a computational sensor for range detection based on light-stripe triangulation. The sensor consists of an array of cells, each cell having both a light detector and a dedicated analog-circuit PE. The light stripe is swept continuously across the scene to be measured. The PE in each cell monitors the output of its associated photoreceptor, recording a time-stamp when the incident intensity peaks. The processing circuitry uses peak detection to identify the stripe and an analog sample-and-hold to record time-stamp data. Each time-stamp fixes the position of the stripe plane as it illuminates the line-of-sight of that cell. The geometry of the projected light stripe is known as a function of time, as is the line-of-sight geometry of all cells. Thus, the 3-D location of the imaged object points ("range pixels") can be determined through triangulation. The cells operate in a completely parallel manner to acquire a frame of 3-D range data, so the spatial resolution of the range

image is determined solely by the size of the array. In the current CMOS implementation, an array of 28 x 32 cells has been fabricated on a 7.9mm x 9.2mm die.

Keast and Sodini [41] at MIT have designed and fabricated a focal plane processor for image acquisition, smoothing, and segmentation. The processor is based on clocked analog CCD/CMOS technology. The light signal is acquired as an accumulated charge. The neighboring PE's share their operands in order to smooth data. In one iteration, each PE sends one quarter of its charge to each of its four neighbors. The charge meets halfway between the pixels and mixes in a single potential well. After mixing, the charge is split in half and returned to the original PE, approximating Gaussian smoothing. However, the segmenting circuit will prevent this mixing if the absolute difference between the neighboring pixels is greater than a given threshold. A 40 x 40 array with a cell size of about 150 x 150 microns is currently being fabricated.

Use of Media Physics (Resistive Grid)

Some algorithms can exploit the physics of the VLSI layers to achieve "processing" in a computational sensor. Carver Mead at Caltech has developed a set of subthreshold CMOS circuits for implementing a variety of vision circuits. The best known design is the "Silicon" retina, a device which computes the spatial and temporal derivative of an image projected onto its phototransistor array. The photoreceptor consists of a phototransistor feeding current into a node of a 48 by 48 element hexagonal resistive grid with uniform resistance values R . The photoreceptor is linked to the grid by a conductance of value G . An amplifier senses the voltage between the receptor output and the network potential. The circuit computes the Laplacian of an image, while temporal derivatives are obtained by adding a capacitor to each node.

Another example which exploits resistive grids to achieve signal processing is the blob position and orientation circuit developed by Standley, Horn, and Wyatt at MIT [83] [84]. Light detectors are placed at the nodes of a rectangular grid made of polysilicon resistors. The photo-current is injected into these nodes and the current flowing out of the perimeter of the grid is monitored. The injected photocurrent and the grid perimeter current are related through

Green's theorem; based on sensed perimeter current, information to compute the first and second moments of the blob is extracted at 5000 frames/sec. An array of 29 x 29 cells has been fabricated on a 9.2mm x 7.9mm die.

3.2 Spatio-Geometric and Optical Computational Sensors

Some computational sensors are based on the "computation" performed by virtue of the special geometry or optical material of the sensor array.

Log-Polar Sensor

The University of Pennsylvania's log-polar sensor developed by Kreider and Van der Spiegel [47] [48] [73] [77] in collaboration with Sandini of University of Genova and researchers at IMEC in Belgium has a radially-varying spatial resolution. A high resolution center is surrounded with a lower resolution periphery in a design resembling a human retina. A sensor that has a high spatial resolution area, like a fovea in a human retina, is often termed a foveating sensor. The image is first mapped from log-polar to the Cartesian plane. There is evidence that in biological systems this type of mapping takes place from eye to brain. The authors have shown that transformations involving perspective, such as optical flow and rotation, are simplified with such a mapping. This sensor must be mechanically foveated for a specific region of interest, and current research concentrates on applying this chip to robotics.

Bederson, Wallace, and Schwartz [7] at New York University and Vision Application, Inc. designed a log-polar sensor as well. The VLSI sensor itself is in the process of being fabricated. An additional interesting part of their system is a miniature pan-tilt actuator called Spherical Pointing Motor (SPM) shown. The SPM is capable of carrying and orienting the sensor. It is an accurate, fast, small, and inexpensive device with low power requirements and is suitable for active vision applications.

Another foveating sensor has been designed by Kosonocky, Wilder and Misra at Rutgers University. The objective was to design a sensor whose foveal region(s) will be able to expand, contract and roam in the field-of-view. The chip is, in essence, a 512x512 square array with the ability to "merge"

its pixels into regions, and output only one value for each such rectangular "super pixel". The largest super pixel is an 8x8 region. There are three modes of operation. In Variable Resolution Mode, the resolution of the entire chip can be selected from highest to lowest, or anywhere inbetween. The Multiple Region of Interest mode provides multiple active windows, possibly with different resolutions, while reading data out from the rest of the array is inhibited. The third mode is a combination of the first two modes. This third mode would resemble the sampling of a human retina if so programmed. The design permits multiple foveae within the retina. The authors demonstrated significant speed-up in data acquisition for a variety of tasks from industrial inspection to target tracking.

Hexagonal Tessellation

Hexagonal sampling tessellates the frequency plane more efficiently than rectangular sampling.¹ Pousart and Trembley [91] at Laval designed a 200 x 200 array with a hexagonal grid. This chip facilitates parallel access to the data in a particular local neighborhood. For rapid convolution, this local neighborhood is subsampled along three principal axes of the grid, thus reducing the data needed for convolution in the local neighborhood of each pixel. Their MAR (Multi-port Array Photo-Receptor system) performs zero-crossing detection at seven spatial frequencies in 16 milliseconds. Edge detection is computed in real time.

Binary Optics

By etching desired geometrical shapes directly into the surface of an optical material, a designer can produce optical elements with properties that were previously impossible to achieve. This method, called binary optics, can perform simple optical processing before the light is detected.

As VLSI microlithographic techniques have advanced, inexpensive fabrication of binary optical devices has become possible [93]. Veldkamp of Lincoln Lab at MIT has developed a micro lens array in which each lens is only 200 microns in diameter. One application of such an array would be to

¹ Nature prefers hexagonal sampling, which is actually found in the mammalian retina.

focus light onto tiny photodetectors thus saving silicon area for processing hardware. Some of the first applications of the idea are already on the market: Hitachi FP-C10 HI-8 video coders use a micro-lens array CCD, and the Sony XC-75 video camera doubles the sensitivity to f8 @ 2000Lux using their HyperHAD CCD structure which uses micro lenses. In addition, binary optics devices have been applied to automatic target recognition and space applications.

McHugh of Hughes Danbury Optical Systems experimented with binary optical techniques and found that they can generate virtually any transformation of an optical wave front. The first application that used this new capability was a binary optical component that optically mapped the log-polar plane to the Cartesian plane. This device, in effect, samples images at log-polar resolution and optically transforms them for sensing on a Cartesian grid. This way an optical log-polar foveating sensor is produced, while the mapping to the Cartesian plane has become "free of charge".

Color and Polarization

Wolff at Johns Hopkins University uses liquid crystal polarizers whose polarization angles are electronically controlled [100]. It has been reported that by eliminating mechanical rotation of filters, switching time between different polarization angles is reduced, and accuracy of results is improved. Wolff hopes to build polarization cameras with polarizers in each element of the CCD array for acquisition of polarized images in real-time. For specular detection, material classification and object recognition, color and polarization carry independent and complementary information: polarization for specularity, and color for diffuse surfaces and light sources. Sensors for real-time combination of both color and polarization images will add rich information to vision systems.

3.3 Computational Modules for Sensory Information Processing

While not strictly a computational "sensor", there is a class of computational modules for sensory information processing which exploit VLSI technologies in a similar manner as computational sensors.

These computational modules are useful when there is not enough space on a single chip to accom-

modate complex PE's, or the data to be processed comes from other modules.

Smoothing and Optimization by Resistive Networks

At Caltech, several regularization techniques have been implemented on-chip. For example, consider the problem of fitting a 2D surface to a set of sparse, noisy depth measurements by imposing a "smoothness" constraint. This method produces quadratically varying functions. This can be solved using simple linear resistive networks by virtue of the fact that the electrical power dissipated in linear networks is quadratic in the current or voltage [71].

Mapping 2D motion algorithms onto analog chips has turned out to be surprisingly difficult. A robust motion detection circuit implemented in analog VLSI has yet to be demonstrated, but early effort has been made by Tanner at Caltech [88] [89]. He successfully built and tested an 8x8 pixel chip that outputs a single uniform velocity averaged over the entire image. His chip reports values of x and y velocity which minimize the least square error in the image brightness constraint equation.

Bair and Koch have successfully built an analog VLSI chip that computes zero crossings of the difference of Gaussians. It takes the difference between two copies of an image, supplied by a 1-D array of 64 photoreceptors, each smoothed by a separate linear first-order resistive network, and reports the zero-crossings in this difference [6]. This implementation has the particular advantage of exploiting the smoothing operation naturally performed by resistive networks, and therefore avoids the burden of additional circuitry. The network resistance and the confidence of the photoreceptor input are independently adjustable for each network. Also, an adjustable threshold on the slope of zero-crossings can be set to cause the chip to ignore weak edges due to noise.

Binary line processes which model discontinuities in intensity within the stochastic framework of Markov Random Fields provide a method to detect discontinuities in motion, intensity, and depth. This is achieved by selectively imposing the smoothness assumption. Harris and Koch have invented the "resistive fuse", which is the first hardware circuit that explicitly implements line processes in a controlled fashion [31]. Like a normal house fuse, a

resistive fuse operates as a linear resistor for small voltage drop and as an open-circuit for large voltage drops. A 20x20 rectangular grid network of fuses has been demonstrated for smoothing and segmenting test images which are scanned onto the chip.

Pyramid

Van der Wal and Burt at David Sarnoff Research Center developed a VLSI pyramid chip PYR [94]. Combined with external framestore, the PYR chip is capable of computing Gaussian and Laplacian pyramid transforms simultaneously. These transforms consist of Gaussian filtering and consecutive subsampling, and, for Laplacian, image subtraction. The Chip has a separable 5 by 5 filter and four 1024-sample-long delay lines. Each filter tap has a preassigned set of possible values. Coefficient values from this set can be changed under software control. PYR has special features such as double precision, double sample density, image border extension and automatic timing control. At 15MHz a single chip can compute Gaussian and Laplacian pyramids at 44 frames/second for 512 by 480 images. PYR is implemented in digital VLSI using the CMOS standard cell library from VLSI Technology, Inc. Digitized image samples pass through the chip sequentially, in raster scan order.

4 Issues

Successful development of a computation sensor relies on careful consideration of several issues including:

- choice of the circuitry: digital vs. analog electronics, choice of sensors with respect to spectral bandwidth (color) and polarizers,
- choice of an algorithm,
- state-of-the-art VLSI,
- prototyping infrastructure: design tools and fabrication facilities,
- applications,
- education, workshops/networking, literature.

All of these issues are discussed in the following sections.

4.1 Analog vs. Digital

Both digital and analog circuits can be implemented using VLSI technology. The analog approach can be conceptually divided into continuous-time (unclocked) and discrete-time (clocked) processing. The choice of technology depends on the particular application, but several general remarks are in order. Compared to digital, the traditional disadvantage of analog electronics is its susceptibility to noise, yielding low precision. The source of this noise can be on-chip switching electronics which require special considerations for hybrid designs. Also, analog electronics do not provide efficient long-term storage; typical storage times are about one second. On the other hand, digital processing requires A/D and D/A conversion, which usually imposes limitations on total circuit speed. Analog electronics are characterized by:

- high speed,
- low latency,
- low precision (typically 6 to 8 bits),
- short data storage time (typically 1 second),
- sensitivity to on-chip digital switching; and
- a long design and testing process.

In general, analog hardware takes less chip area than digital mechanisms of the same functionality. Most participants at the workshop were experts in analog circuitry which seems to be preferred; however, many recognized the importance of digital electronics for computational sensing.

Analog VLSI offers two interesting advantages for computational sensor design. First, the physical properties of the solid-state layers and devices can sometimes be exploited to yield elegant, new solutions. One such example is to exploit the physics of a resistive sheet (or dense grid) to compute desired quantities.

The second interesting advantage of analog VLSI is charge-domain processing, best exemplified by CCD technology, which offers an area-efficient mechanism for transferring data. In addition, creative processing schemes can be developed to process the data in charge-domain as it is transferred. CCD technology has already provided several useful examples of integrated sensing and signal processing.

4.2 Algorithms for Computational Sensors

While the VLSI computational sensor offers exciting opportunities, one must be careful in deciding which algorithms or applications will benefit from such an implementation. At the present state of technology, successful design of working VLSI circuits, especially analog ones, is a lengthy process.

Algorithms must be carefully selected or invented to match the architecture to the circuitry for maximum performance - there are definite limitations on circuitry and architectures. Circuitry has limited precision and storage. Until technology allows much denser circuits (or 3D structures) for example, there is not enough room to fabricate a complex PE at each photo site.

Simple cell-parallel algorithms that detect local cues or integrate local information over time or multiple channels (eg. spectrum) at each cell are most ideal.

When a complex PE is required, processing and sensing can take place on separate, but tightly coupled (preferably on-chip) modules. The cost of transferring data must be minimized in order to justify the use of VLSI over conventional computer systems. CCD row-parallel transfer is one way to perform the transfer at a reasonable speed. Also, some algorithms do not directly exhibit parallelism in the focal plane: they often require significant local data storage at each PE. In stereo algorithms, for example, optical signals are to be combined from two different focal planes. In this case, data are read out and processed on a separate computational module.

There are optimizations and other techniques that map naturally to physical processes in silicon; such as relaxation processes implemented on resistive grids. The advantage of these physics-based processors over computer implementation is that they minimize a multi-dimensional energy function by reaching a stable state of a continuous-time system, potentially reducing round-off error and numerical instability from which an iterative solution by a digital computer may suffer.

In summary, the following are some general characteristics of algorithms which are good candidates for computational sensors implementation:

- Algorithms that are simple and robust to noise, and are based on sensor or cue integration

- Algorithms that exploit a significant level of parallelism without requiring significant storage capacity, wafer real estate, or inter-processor data transfer.
- Algorithms that map naturally to physical processes encountered in semiconductors,
- Algorithms that could exploit the intercommunication and propagation afforded by charge-transfer, surface acoustic waves, and optical properties.

4.3 VLSI Technology

CMOS, Bipolar, and BiCMOS are the most available VLSI technologies. CMOS is characterized by very dense packaging, low power consumption, and high input impedance. Good switching properties make it well suited for digital, switching, and hybrid circuits. It is widely accessible and relatively inexpensive technology. CCD's are implemented in MOS technology.

Bipolar technology is characterized by low noise and fast circuitry, but consumes more power and takes more substrate real estate. It is not as accessible to the wider research community as it probably should be.

BiCMOS combines the advantages of both CMOS and Bipolar technologies.

Semiconductor material other than silicon is also available. GaAs compounds yield very high speed circuitry and are well suited to electro-optical applications. GaAs technology is less available, however, and is considerably more expensive.

The trend in VLSI is toward smaller device geometries. This produces both smaller and faster digital circuits and hence more functionality per unit area. This scaling, however, is not as beneficial to analog circuitry as to digital. Most active devices are designed at a given size and scaling and would not preserve desired functional features after a scale change. Analog MOS circuits benefit more from improvements in fabrication process quality. Factors such as oxide quality and thickness, or tighter control of threshold voltages would greatly benefit analog circuit performance.

Great interest has been shown in 3D VLSI. One possibility is optical signal communication between stacked chips. This could be accomplished with

the availability of silicon-compatible semiconductor emitters and IR detectors [90]. This technique would also require and exploit integrated optics capability such as binary optics. Alternatively, a conducting feedthrough could be developed for making distributed point-to-point electrical connections [70].

Micro fiber-optics could be used to route data in parallel from module to module. The optical approach has the advantage of possible optical processing during the data transmission itself, but has the disadvantage of high power consumption and heat dissipation. This technology has not been developed far enough to become accessible to the wider research community.

4.4 Applications

As VLSI technology advances and becomes accessible to a wider research community, a number of ideas that combine sensing and processing on a chip are emerging. Many attempts, however, are too quick to postulate miraculous chips and systems which have little chance of ever working.

Several successful examples of computational sensors have been driven by applications, and the workshop participants have agreed that this will remain true for most successful developments. A truly successful "marriage" of sensing and computation can be done only by careful analysis of application requirements in conjunction with implementation technologies.

While a wide variety of applications are conceivable, the following are potential applications that have been suggested during the workshop:

- A high resolution camera (2000 x 2000 and up).
- Face recognition for credit purchase, security, and human-computer interfaces.
- An inexpensive anti-collision stereo sensor for automobiles.
- Motion detection and tracking for automobiles, security, and human-computer interfaces.
- Automatic local brightness adjustment of images.
- Tactile sensors for material handling.

- Insect robots for the toy industry.
- High-speed industrial inspection, chip reticle alignment.
- Document understanding and optical character recognition.
- A light-weight amacronic sensor/display device for virtual reality.
- Image compression for home video appliances.
- Medical sensors/implants.
- Automatic target recognition - signal preprocessing for specialized sensors (gain, bias, filtering) and multi-sensor integration. An example is a computational sensor to perform the functions of detection, inscan calibration, and output multiplexing of FLIR.
- Space robotics for orbital replacement, satellite retrieval, and planetary exploration.
- Remotely and automatically piloted vehicles - sensors to make UGV, AAV, AUV low cost.

4.5 Prototyping Infrastructure

Design tools

An issue which received unanimous agreement among workshop participants is the lack of analog VLSI design tools equivalent to those for digital design. These tools include design aids from layout to testing, including extraction, verification and simulation. Analog circuits are more sensitive to parasitics than digital circuits. Accurate techniques for including these parasitics in the extracted files would reduce the number of design iterations due to unexpected circuit behavior.

Analog modeling and simulation capabilities are still inadequate. Much of the attention in modeling is directed at the effects of extremely short channel lengths on MOS transistor operation. Analog design rarely uses minimum size transistors, but is more critically dependent upon operating under a different bias condition: subthreshold and saturation regions. The proper modeling of bias-dependent capacitances is critical for modeling circuit dynamics and stability. There is little or no support for simulating charge-domain devices like CCD's. Statistical

modeling is an important predictive element of analog design, providing assurance that the resulting circuits will meet the prescribed design constraints. Without it, a circuit may be functional and within specifications for a given process model, but actual process variation may result in an out-of-spec or inoperable circuit.

It has been noted that a data book for standard analog cells would be very useful. While it will be more difficult than the digital domain, it is necessary to develop a library of standard building blocks of compatible electronic and sensor components with which one can design a new computational sensor.

Fabrication Facilities

The MOSIS Service is a prototyping service offering fast-turnaround standard cell and full-custom VLSI circuit development at very low cost. The MOSIS Service, begun in 1980, provides fabrication services to government contractors, agencies, and university classes under the sponsorship of the Defense Advanced Research Projects Agency (DARPA) with assistance from the National Science Foundation (NSF). MOSIS has developed a methodology that allows the merging of many different projects from various organizations onto a single wafer. Instead of paying for the cost of mask-making, fabrication, and packaging for a complete run (currently between \$50,000 and \$80,000) MOSIS users pay only for the fraction of the silicon that they use, which can cost as little as \$400. Initially, the MOSIS user-base was primarily university and government users. MOSIS' success in serving this group of users led, in recent years, to a natural expansion into the industrial sector, with rapidly growing use of MOSIS by commercial companies. MOSIS foundries have also taken advantage of the frequent prototype runs for their own needs as well as those of their clients. MOSIS is located at the Information Sciences Institute of the University of Southern California (USC/ISI) in Marina del Rey, California.

The MOSIS program has been a successful mechanism for promoting VLSI applications. MOSIS' ease of access, quick turnaround, and cost-effectiveness have afforded designers opportunities for frequent prototype iterations that otherwise might not even have been considered. With MOSIS' low cost for "tiny-chip" fabrication, silicon can be

used as a rapid prototyping vehicle. Small functional building blocks can be easily fabricated and tested before too much time is invested in building and integrating a full system. Furthermore, many ideas and needed intuition can be gained through "playing" with these actual working chips. Successful designers of existing functional computational sensors have reported that silicon prototyping, combined with higher level algorithm simulation, has proven to be a useful system-building approach in computational sensors.

MOSIS offers two monthly runs of a standard 2 μ m, double-layer metal, CMOS process. One of these runs usually includes a second layer of polysilicon. Typically these designs are fabricated, bonded and returned in about two months. In addition to these standard runs, a 1.2 μ m CMOS run goes out about once every month and there are more infrequent runs at 0.8 μ m. Every other month includes a low-noise 2 μ m analog CMOS run which has options for second poly, a NPN bipolar transistor in the n-well, and a buried channel CCD.

MOSIS's capability, however, is limited for the research and development of computational sensors. Quality bipolar and depletion-mode MOS devices are unavailable. MOSIS is beginning to offer GaAs (instead of the more usual Silicon) process runs on a regular basis.

At this point, MOSIS does not provide a capability for optical electronics fabrication. University researchers must rely on teaming with industries which have the fabrication capability in this area. It is noteworthy that both the European research community and the Japanese micro-sensor project will have a common facilities including capabilities for optical electronics fabrication.

4.6 Education, Workshops/Networking and Literature

Understanding semiconductor and device physics as well as techniques for marketing custom-made integrated circuits are essential prerequisites to developing a successful computational sensor. For the complete success of a computational sensor, avenues of communication between VLSI designers, computer vision researchers, and product developers must be developed. These groups would exchange information about the opportunities and difficulties in each others' fields. Vision (and other

sensor) researchers must be made aware of what is available in VLSI technology, and VLSI designers must understand the problems of machine vision. This workshop was very productive. It was recommended that follow-on workshops or conferences be held.

It was proposed that universities and industries team-up to allow students to obtain more hands-on experience. This is an old idea that still has difficulty working in practice. Namely, most students and university professors are more likely to undertake theoretical research than to work on the "real thing". This is primarily due to the fact that dealing with hardware tends to extend time in graduate school for students, and reduce the publishing rate of professors. This problem received some attention, and reviews of academic standards were suggested. It was suggested that more credit should be given to efforts which produce working prototype devices or systems.

The body of experience and knowledge of computational sensors is currently scattered over a large number of disciplines and corresponding publications. Publications range from journals on electronic circuits and signal processing to publications on neural networks and vision research. To effectively communicate knowledge about computational sensors, it was suggested that a new journal be created.

Another type of cooperation is to distribute working prototype sensors in among the user community. An excellent example is the log-polar camera prototype that University of Pennsylvania has offered to share with interested researchers. This type of cooperation is of mutual benefit to the sensor designers as well as to application developers. Designers of the computational sensor receive much needed feedback about the actual need and practical value of the sensor, while application researchers can investigate new areas previously limited by the absence of these specialized devices.

5 Recommendations

In light of the previous analysis, the workshop has recommended the following:

1. Create a research and development program in computational sensors. The program must have the following characteristics (Figure 1):

- Interdisciplinary - the program must include sensing, algorithms, VLSI, material, and applications;
- Multi-modal - the program must deal with not only the image or visual modality, but also with other sensing modalities including tactile, acoustic, pressure, acceleration, chemical, and so on;
- Prototyping-oriented - individual projects under this program must be oriented toward producing working prototype devices or systems;
- Applications - individual projects must identify potential applications and possible avenues of technology transfer to real world applications.

2. Improve the infrastructure for research and development of computational sensors:

- Fabrication facilities - MOSIS (or similar facilities) must be expanded to include technologies for optical and mechanical sensor development;
- Tools - Tools for designing and testing computational sensors can be far more complicated, than they are for standard VLSI design. Standardization, and library and tool development are essential;
- Education - Hands-on experience must be provided to graduate students;
- Networking and workshops - Researchers in computational sensors, by its nature, are scattered in multiple fields, and mechanisms; workshops and consortiums must be developed to bring them together.

THE FOLLOWING BIBLIOGRAPHY CONTAINS PAPERS COLLECTED DURING AND AFTER THE WORKSHOP BY THE CONTRIBUTIONS OF PARTICIPANTS.

References

- [1] A. Andreou, K. Strohhahn, and R.E. Jenkins, "Silicon Retina for Motion Computation", Proc. IEEE Int. Symp. Circuits and Systems, pp. 1373-1376, 1991.

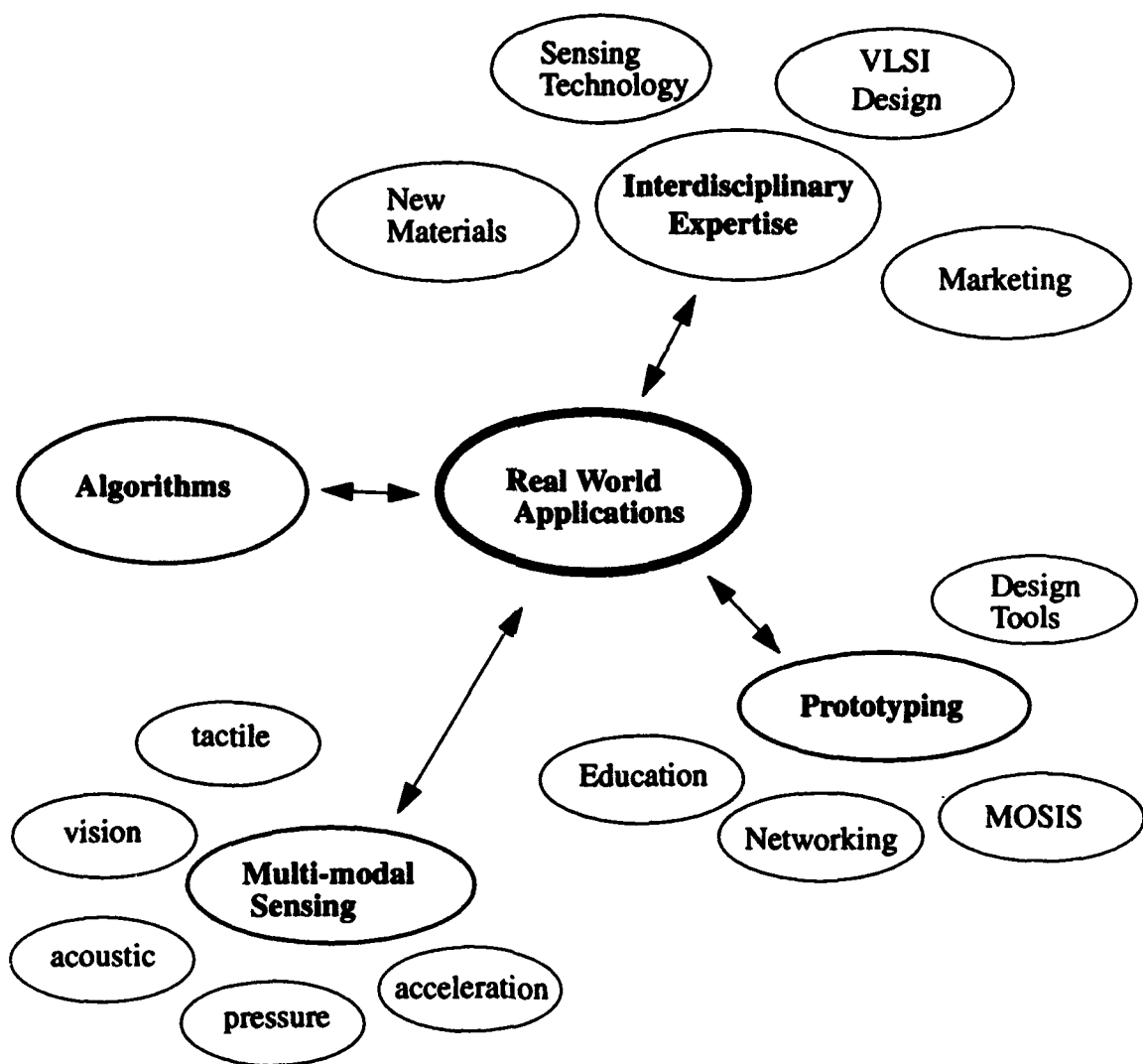


Figure 1: Computational Sensor Program

- [2] A.G. Andreou, K.A. Boahen, A. Pavasovic, P.O. Pouliquen, R.E. Jenkins and K. Strohbehn, "Current-Mode Subthreshold MOS Circuits for Analog VLSI Neural Systems," *IEEE Transactions on Neural Networks*, Vol. 2, No. 2, pp. 205-213, March 1992
- [3] A.G. Andreou and K.A. Boahen, "Synthetic Neural Circuits Using Current-Domain Signal Representations," *Neural Computation*, Vol. 1, No. 4, pp. 489-501, 1989.
- [4] A.G. Andreou and C.R. Westhate, "The Magnetotransistor Effect," *IEE Electronics Letters*, Vol. 20, No. 17, pp. 699-701, August 1984.
- [5] A.G. Andreou, "Electronic Receptors for Tactile/haptic Sensing," *Advances in Neural Information Processing Systems*, Vol. 1, edited by D.S. Touretzky, Morgan Kaufmann Publishers, San Mateo, pp. 785-792, 1989.
- [6] W. Bair and C. Koch, "An Analog VLSI Chip for Finding Edges from Zero-crossings," In: *Advances in Neural Information Processing Systems*, R. Lippman, J. Moody, and D.S. Touretzky, eds., Vol. 3, pp. 399-405, Morgan Kaufmann, San Mateo, CA, 1991.
- [7] Bederson, R. Wallace and E. Schwartz, "A Miniature Pan-Tilt Actuator: the Spherical Pointing Motor," Tech. Rep. No. 601, Courant Institute of Mathematical Sciences, New York University, March 1992.
- [8] B.E. Boser, et. al., "An Analog Neural Network Processor with Programmable Topology," *IEEE JSSC*, Vol. 26, No. 12, pp. 2017-2025, December 1991.
- [9] William O. Camp, Jr., Jan Van der Spiegel and Min Xiao, "A Line and Edge Orientation Sensor," presented at the 1992 IJCNN Conference, Baltimore, MD, June 1992.
- [10] L.R. Carley's Structured light Depth Sensor Chip
- [11] L.R. Carley and T. Kanade, "DARPA Semiannual Report," January 1992.
- [12] A.M. Chiang and M.L. Chuang, "A CCD Programmable Image Processor and Its Neural Network Applications," *IEEE JSSC*, Vol. 26, No. 12, pp. 1894-1901, December 1991.
- [13] C. P. Chong, A.T. Salama and K.C. Smith, "Image-Motion Detection Using Analog VLSI," *IEEE Jour. Solid-State Circuits*, Vol. 27, No. 1, pp. 93-96, January 1992.
- [14] L.O. Chua and L. Yang, "Cellular Neural Networks: Theory," *Trans. Circuits and Systems*, Vol. 35, No. 10, pp. 1257-1272, October 1988.
- [15] L.O. Chua and L. Yang, "Cellular Neural Networks: Applications," *Trans. Circuits and Systems*, Vol. 35, No. 10, pp. 1273-1290, October 1988.
- [16] L.O. Chua, L. Yang and K.R. Krieg, "Signal Processing Using Cellular Neural Networks," *Jour. VLSI Signal Processing*, Vol. 3, Nos. 1 and 2, pp. 25-51, June 1991, and the references therein.
- [17] M.H. Cohen and A.G. Andreou, "MOS Circuit for Nonlinear Hebbian Learning," *IEE Electronics Letters*, Vol. 28, No. 6, pp. 591-593, March 1992.
- [18] R.R. Etienne-Cummings, S.A. Fernando and J. Van der Spiegel, "Real-Time 2-D Analog Motion Detector VLSI Circuit," *Proceedings of the 1992 IJCNN Conf.*, Baltimore, MD, June 7-11, 1992.
- [19] S.J. Decker, "A Resistive Fuse for Image Segmentation and Smoothing," S.M. Thesis, Department of Electrical Engineering and Computer Science, MIT, August 1991.
- [20] T. Delbruck, "A Chip that Focusses an Image on Itself," in *Analog VLSI Implementation of Neural Systems*, C. Mead and M. Ismail, eds., Kluwer, Boston, pp. 171-188, 1989.
- [21] T. Delbruck and C. Mead "An Electronic Photoreceptor Sensitive to Small Changes in Intensity," in *Neural Information Processing Systems 1*, D. Touretzky, ed., pp. 720-727, Morgan Kaufmann, San Mateo, CA, 1989.
- [22] J. Dominguez, "Effortless Internal Camera Calibration Using the Images of Spheres," S.B. Thesis, Department of Electrical Engineering and Computer Science, MIT, May 1991.
- [23] L. Dron, "The Multi-Scale Veto Model: A Two-Stage Analog Network for Edge Detection and Image Reconstruction," submitted to *Int'l Jour. Computer Vision*, 1992.
- [24] L. Dron, "An Analog Model of Early Visual Processing: Contour and Boundary Detection in the Retina," *SPIE, Intelligent Robots and Computer Vision X*, Boston, MA, November 14-15, 1991. (Proceedings to appear.)
- [25] C.L. Fennema and W.B. Thompson "Velocity Determination in Scenes Containing Several Moving Objects," *Comp. Vis. Graph. Proc.* No. 9, pp. 301-315, 1979.
- [26] A. Gruss, L.R. Carley and T. Kanade, "Integrated Sensor and Range-Finding Analog Signal Processor," *Jour. Solid-State Circuits*, Vol. 26, No. 3, pp. 184-191, March 1991.
- [27] A. Gruss, S. Tada and T. Kanade, "A VLSI Smart Sensor for Fast Range Imaging," *Proc IEEE International Conference on Intelligent Robots and Systems (IROS'92)*, Raleigh NC, July 7-10, 1992.
- [28] M. Hakkarainen, J. Little, H.-S. Lee and J.L. Wyatt, Jr., "Interaction of Algorithm and Implementation for Analog VLSI Stereo Vision," *SPIE Int'l. Symp. on Optical Eng. and Photonics in Aerospace Sensing*, Orlando, FL, pp. 173-184, April, 1991.

- [29] M. Hakkarainen and H.-S. Lee, "A Stereo Vision System in CCD/CMOS Technology," submitted to 1992 European Solid-State Circuits Conference, Copenhagen, Denmark.
- [30] M. Hakkarainen, "A Real-Time Stereo Vision System in CCD/CMOS Technology," Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, May 1992.
- [31] J.G. Harris, C. Koch and J. Luo, "A two-dimensional analog VLSI circuit for detecting discontinuities," *Science*, Vol. 248, pp 1209-1211, 1990.
- [32] J. Harris, S.C. Liu and B. Mathur, "Discarding Outliers Using a Nonlinear Resistive Network," *Proc. IJCNN*, Vol. 1, pp. 501-506, Seattle, Wash., July 1991.
- [33] J.A. Hengeveld, "Smooth Surface Reconstruction in VLSI," S.B. Thesis, Department of Electrical Engineering and Computer Science, MIT, May 1989.
- [34] T. Horiuchi, J. Lazzaro, A. Moore, C. Koch "A Delay-line Based Motion Detection Chip," in *Advances in Neural Information Processing Systems* Vol. 3, R. Lippman, J. Moody, D. Touretzky, eds., pp. 406-412, Morgan Kaufmann, San Mateo, CA 1991.
- [35] B.K.P. Horn, "Parallel Networks for Machine Vision," A.I. Memo No. 1071, MIT, December 1988.
- [36] B. Horn, H.-S. Lee, C. Sodini, T. Poggio and J. Wyatt, "The First Three Years of the MIT Vision Chip Project - Analog VLSI Systems for Integrated Image Acquisition and Early Vision Processing," VLSI Memo 91-645, MIT, October 1991.
- [37] R.E. Howard, et. al., "Optical Character Recognition: A Technology Driver for Neural Networks," *Proc. 1990 IEEE Int. Symp. on Circuits and Systems*, pp. 2433-2436, New Orleans, LA, May 1990, and the references therein.
- [38] M. Ishikawa, A. Morita and N. Takayanagi, "High Speed Vision System Using Massively Parallel Processing," submitted to IROS '92.
- [39] T. Kanade, A. Gruss and L.R. Carley, "A Very Fast VLSI Rangefinder," *Proc. IEEE International Conference on Robotics and Automation*, pp. 1322-1329, Sacramento, CA, April 1991.
- [40] C.L. Keast and C.G. Sodini, "A CCD/CMOS Process for Integrated Image Acquisition and Early Vision Signal Processing," *Proc. SPIE Charge-Coupled Devices and Solid State Sensors*, Santa Clara, CA, pp. 152-161, February 1990.
- [41] C.L. Keast and C.G. Sodini, "An Integrated Image Acquisition, Smoothing and Segmentation Focal Plane Processor," to appear in 1992 VLSI Circuit Symposium, Seattle, WA, June 1992.
- [42] H. Kobayashi, J.L. White and A.A. Abidi, "An Active Resistor Network for Gaussian Filtering of Images," *Jour. Solid-State Circuits*, Vol. 26, No. 5, pp. 738-748, May 1991.
- [43] C. Koch, "Seeing Chips: Analog VLSI Circuits for Computer Vision," *Neural Computation*, No. 1, pp. 184-200, 1989.
- [44] C. Koch, H.T. Wang, B. Mathur, A. Hsu and H. Suarez, "Computing Optical Flow in Resistive Networks and the Primate Visual System," *Proc. IEEE Workshop on Visual Motion*, pp. 62-72, Irvine, CA, March 1989.
- [45] C. Koch, W. Bair, J.G. Harris, T. Horiuchi, A. Hsu and J. Luo, "Real-Time Computer Vision and Robotics Using Analog VLSI Circuits," in *Advances in Neural Information Processing Systems* Vol. 2., D. Touretzky, ed., pp. 750-757, Morgan Kaufmann, San Mateo, CA, 1990.
- [46] C. Koch, A. Moore, W. Bair, T. Horiuchi, B. Bishofberger and J. Lazzaro, "Computing Motion Using Analog VLSI Vision Chips: an Experimental Comparison Among Four Approaches," *Proc. IEEE Workshop on Visual Motion*, pp. 312-324, Princeton, NJ, October 1991.
- [47] G. Kreider, J. Van der Spiegel, I. Born, C. Claeys, I. Debusschere, S. Sandini, P. Dario and F. Fantini, "A Retina-Like Space Variant CCD Sensor," *Proc. SPIE Conf. on Charge-Coupled Devices and Solid State Optical Sensors*, Vol. 1242, pp. 133-140, Santa Clara, CA, Feb. 12-13, 1990.
- [48] G. Kreider, J. Van der Spiegel, I. Born, C. Claeys, I. Debusschere, G. Sandini and P. Dario, "The Design and Characterization of a Space Variant CCD Sensor," *SPIE Conf. on Intelligent Robots and Computer Vision IX: Algorithms and Techniques*, Vol. 1381, pp. 242-249, Boston, Nov. 1990.
- [49] J. Lazzaro and C. Mead, "Circuit Models of Sensory Transduction in the Cochlea," *Analog VLSI Implementations of Neural Networks*, C. Mead and M. Ismail, eds., pp. 85-101, Kluwer, Norwell, MA, 1989.
- [50] J. Lazzaro, S. Ryckebusch, M.A. Mahowald and C. Mead, "Winner-Take-All Networks of $O(n)$ Complexity," *Advances in Neural Information Processing Systems* Vol. 1, D. Touretzky, ed., pp. 703-711, Morgan Kaufmann, San Mateo, CA, 1988.
- [51] H.S. Lee, and P. Yu, "CMOS Resistive Fuse Circuits," 1991 Symp. on VLSI Circuits, pp. 109-110, Oiso, Japan, May 1991.
- [52] S. Liu and J.G. Harris, "The Negative Fuse Network," *Visual Information Processing: From Neurons to Chips*, SPIE Proceedings, Vol. 1473, pp. 185-193, Orlando, FL, April 1991.

- [53] W. Liu, A.G. Andreou and M.H. Goldstein, "Voiced-Speech Representation by an Analog Silicon Model of the Auditory Periphery," IEEE Transactions on Neural Networks, Vol. 3, No. 3, pp. 477-487, May 1992.
- [54] J. Lumsdaine, A. Wyatt, Jr. and I. Elfadel, "Parallel Distributed Networks for Image Smoothing and Segmentation in Analog VLSI," Proc. 28th IEEE Conf. on Decision and Control, pp. 272-279, Tampa, FL, December 1989.
- [55] A. Lumsdaine, J.L. Wyatt, Jr., and I.M. Elfadel, "Non-linear Analog Networks for Image Smoothing and Segmentation," Journal of VLSI Signal Processing, Vol. 3, pp. 53-68, 1991.
- [56] M. Mahowald, "Silicon Retina with Adaptive Photoreceptors," Visual Information Processing: From Neurons to Chips, SPIE Proceedings, Vol. 1473, pp. 52-58, Orlando, FL, April 1991.
- [57] M.A. Mahowald and T. Delbruck, "Cooperative Stereo Matching Using Static and Dynamic Image Features," in Analog VLSI Implementation of Neural Systems, C. Mead and M. Ismail, eds., Kluwer, Boston, pp. 213-238, 1989.
- [58] M.A. Mahowald and C. Mead, "Silicon Retina," Chap. 15 of Analog VLSI and Neural Systems, Addison-Wesley Publishing Co., 1989.
- [59] J. Mann, "Implementing Early Visual Processing in Analog VLSI: Light Adaptation," Visual Information Processing: From Neurons to Chips, SPIE Proceedings, Vol. 1473, pp. 128-136, Orlando, FL, April 1991.
- [60] L. Massone, T. Sandini, V. Tagliasco, "Form-invariant Topological Mapping Strategy for 2D Shape recognition," Comp. Graphics and Image Processing, Vol. 30, pp. 1169-1188, 1985.
- [61] B. Mathur and C. Koch, "Visual Information Processing: >From Neurons to Chips," eds., Proc. SPIE, Vol. 1473, 1991.
- [62] T. Matsumoto, L.O. Chua and H. Suzuki, "CNN Cloning Template: Connected Component Detector," Trans. Circuits and Systems, Vol. 37, No. 5, pp. 633-635, May 1990.
- [63] T. Matsumoto, L.O. Chua and R. Furukawa, "CNN Cloning Template: Hole Filler," Trans. Circuits and Systems, Vol. 37, No. 5, pp. 635-638.
- [64] T. Matsumoto, L.O. Chua and T. Yokoharna, "Image Thinning with a Cellular Neural Network," Trans. Circuits and Systems, Vol. 37, No. 5, pp. 638-640, May 1990.
- [65] I.S. McQuirk, "Direct Methods for Estimating the Focus of Expansion in Analog VLSI," S.M. Thesis, Department of Electrical Engineering and Computer Science, MIT, September 1991.
- [66] C. Mead, "A sensitive electronic photoreceptor," in 1985 Chapel Hill Conference on Very Large Scale Integration, H. Fuchs, ed., Computer Science Press, Chapel Hill, NC, 1985.
- [67] C. Mead, "Adaptive Retina," Analog VLSI Implementation of Neural Systems, C. Mead and M. Ismail, eds., Kluwer, Boston, pp. 239-246, 1989.
- [68] C. Mead, "Analog VLSI and Neural Systems," Reading, MA: Addison-Wesley, 1989.
- [69] A. Moore and C. Koch, "A multiplication-based analog motion detection chip," Visual Information Processing: From Neurons to Chips, B. Mathur and C. Koch, eds., Proc. SPIE, Vol. 1473, pp. 66-75, 1991.
- [70] T. Nishimura, Y. inoue, K. Sugahara, S. Kusunoki, T. Kumamoto, S. Nakagawa, M. Nakaya, Y. Horiba and Y. Akasaka, "Three Dimensional IC for High Performance Image signal Processor," Proc. of the IEDM - Int. Elec. Dev. Meeting, Washington, D.C. December 6-9, 1987.
- [71] T. Poggio and C. Koch, "Ill-posed problems in early vision: from computational theory to analogue networks," Proc. R. Soc. Lond. Vol. B225, pp. 303-323, 1985.
- [72] T. Poggio, W. Yang and V. Torre, "Optical flow: computational properties and networks, biological and analog," The Computing Neuron, R. Durbin, C. Mian and G. Mitchison, eds., pp. 355-370. Addison-Wesley, Menlo Park, CA, 1989.
- [73] G. Sandini, V. Tagliasco, "An Anthropomorphic Retina-like Structure for Scene Analysis," Comp. Graphics and Image Processing, Vol. 14, pp. 365-372, 1980.
- [74] R. Sarpeshkar, J.L. Wyatt, Jr., N.C. Lu and P.D. Gerber, "Mismatch Sensitivity of a Simultaneously Latched CMOS Sense Amplifier," IEEE Jour. Solid-State Circuits, Vol. 26, No. 10, pp. 1413-1422, October 1991.
- [75] M.N. Seidel, J.L. Wyatt, Jr., "Using-Time Bounds for Switched Capacitor Networks," Proc. IMACS World Congress on Computation and Applied Mathematics, Dublin, Ireland, pp. 1669-1670, July 1991.
- [76] L.M. Silveira, A. Lumsdaine and J. White, "Parallel Simulation Algorithms for Grid-Based Analog Signal Processors," Proc. IEEE Int'l. Conf. on CAD, Santa Clara, CA, pp. 442-445, November 1990.
- [77] J. Van der Spiegel, G. Kreider, C. Claeys, I. Debusschere, G. Sandini, P. Dario, F. Fantini, P. Bellutti and G. Soncini, "A Foveated Retina-Like Sensor Using CCD Technology," Chap. 8 of Analog VLSI Implementation of Neural Systems, C. Mead and M. Ismail, eds., Kluwer Academic Publishers, 1989.
- [78] D.L. Standley, "Criteria for Robust Stability in a Class of Lateral Inhibition Related Networks Coupled

- Through a Resistive Grid*," Proc. 1989 Conf. on Information Sciences and Systems, Johns Hopkins University, Baltimore, MD, pp. 416, March 1989.
- [79] D. Standley, "Stability in a Class of Resistive Grid Networks Containing Active Device Realizations of Non-linear Resistors," Proc. IEEE Int'l. Symp. on Circuits and Systems, pp. 1474-1477, New Orleans, LA, May 1990.
 - [80] Standley, D.L., "Design Criterion Extension for Stable Lateral Inhibition Networks in the Presence of Circuit Parasitics," Proc. IEEE Symp. on Circuits and Systems, pp. 837-840, Portland, OR, May 1989.
 - [81] D. Standley, "An Object Position and Orientation IC with Embedded Imager," IEEE Journal of Solid-State Circuits, Vol. 26, No. 12, pp. 1853-1860, 1991.
 - [82] D.L. Standley, "Analog VLSI Implementation of Smart Vision Sensors: Stability Theory and an Experimental Design," Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, January 1991.
 - [83] D. Standley, B. Horn, "An Object Position and Orientation IC with Embedded Imager," Proc. IEEE Int'l. Solid State Circuits Conf., San Francisco, CA, pp. 38-39, February 13-15, 1991.
 - [84] D. Standley, and B.K.P. Horn, "Analog CMOS IC for Object Position and Orientation," SPIE Int'l. Symp. on Optical Eng. and Photonics in Aerospace Sensing, Orlando, FL, pp. 194-201, April 1991.
 - [85] D.L. Standley and J.L. Wyatt, Jr., "Circuit Design Criteria for Stability In A Class of Lateral Inhibition Neural Networks," Proc. 27th Conf. on Decision and Control, Austin, TX, pp. 328-333, December 1988.
 - [86] D.L. Standley, and J.L. Wyatt, Jr., "Stability Criterion for Lateral Inhibition and Related Networking that is Robust in the Presence of Integrated Circuit Parasitics," IEEE Trans. Circuits and Systems, Vol. 36, No. 5, pp. 67-81, May 1989.
 - [87] J.E. Tanner and J. Luo, "A Single Chip Imager and Feature Extractor," Visual information processing: From Neurons to Chips, Proc. SPIE, Vol. 1473, Orlando, FL, pp. 76-87, April 1991.
 - [88] J. Tanner and C. Mead, "An integrated optical motion sensor," VLSI Signal Processing II, S-Y. Kung, R.E. Owen, and J.G. Nash, eds., pp. 59-76, IEEE Press, NY, 1986.
 - [89] J. Tanner and C. Mead, "Optical motion sensor," in Analog VLSI and Neural Systems, C. Mead, pp. 229-255, Addison-Wesley, Reading, MA, 1989.
 - [90] G.W. Turner, C.K. Chen, B.-Y. Tsaur and A.M. Waxman, "Through-Wafer Optical Communication Using Monolithic InGaAs-on-Si Led's and Monolithic PtSi-Si Schottky-Barrier Detectors," IEEE Photonics Technology Letters, Vol. 3, No. 8, August 1991.
 - [91] M. Tremblay and D. Poussart, "Focal Plane VLSI Processing for Multiresolution Edge Extraction," SPIE Aerospace Sensing, Orlando, 1992.
 - [92] C.B. Umminger and C.G. Sodini, "Switched Capacitor Networks for Focal Plane Image Processing Systems," to appear in IEEE Trans. Circuits and Systems for Video Technology, September 1992.
 - [93] W. B. Veldkamp and T. J. McHugh, "Binary Optics," Scientific American, May 1992.
 - [94] G.S. Van der Wal and P.J. Burt, "A VLSI Pyramic Chip for Multiresolution Image Analysis," to appear in The International Journal of Computer Vision.
 - [95] C.F.R. Weiman, "Exponential Sensor Array Geometry and Simulation," Proc. SPIE Conf. on Pattern Recognition and Signal Processing, Vol. 938, Digital and Optical Shape Representation and Pattern Recognition, pp. 129-137, Orlando, FL, 1988.
 - [96] C.F.R. Weiman, "3-D Sensing with exponential sensor arrays," Proc. SPIE Conf. on Pattern Recognition and Signal Processing, Vol. 938, Digital and Optical Shape Representation and Pattern Recognition, Orlando, FL, 1988.
 - [97] C.F.R. Weiman, "Polar Exponential Sensor Arrays Unify Iconic and Hough Space Representation," Proc. SPIE Conf. on Intelligent Robots and Computer Vision VIII: Algorithms and Techniques, Vol. 1192, Philadelphia, PA, pp. 832-842, November 1989.
 - [98] C.F.R. Weiman and R.D. Juday, "Tracking Algorithms Using Log-Polar Mapped Image Coordinates," Proc. SPIE Conf. on Intelligent Robots and Computer Vision VIII: Algorithms and Techniques, Vol. 1192, Philadelphia, PA, November 1989.
 - [99] C.F.R. Weiman and G. Chaikin, "Logarithmic Spiral Grids for Image Processing and Display," Computer Graphics and Image Processing, Vol. 11, pp. 197-226, 1979.
 - [100] L. B. Wolff and T.A. Mancini, "Liquid Crystal Polarization Camera," Computer Science Technical Report 92-09, John Hopkins University, Baltimore, MD, May 1992.
 - [101] J. L. Wyatt, Jr. and D.L. Standley, "Criteria for Robust Stability in a Class of Lateral Inhibition Networks Coupled Through Resistive Grids," Neural Computation, Vol. 1, No. 1, pp. 58-67, Spring 1989.
 - [102] J.L. Wyatt, Jr., C. Keast, M. Seidel, D. Standley, B. Horn, T. Knight, C. Sodini, H.-S. Lee and T. Poggio, "Analog VLSI Systems for Early Vision Processing," Proc. 1992 IEEE Int'l. Symp. on Circuits and Systems, pp. 1644-1647, San Diego, CA, May 1992.
 - [103] J.L. Wyatt Jr., D. Standley and B. Horn, "Local Computation of Useful Global Quantities Using Linear Resistive-Grid Networks," poster session, Conf. on

Neural Networks for Computing, Snowbird, UT, April 1990.

- [104] J.L. Wyatt, Jr., D.L. Standley, and W. Yang, "*The MIT Vision Chip Project: Analog VLSI Systems for Fast Image Acquisition and Early Vision Processing*," in Proc. 1991 IEEE Int'l. Conf. on Robotics and Automation, pp. 1330-1335, Sacramento, CA, April 1991.
- [105] W. Yang, "*A Charged-Coupled Device Architecture for On Focal Plane Image Signal Processing*," Proc. Int'l. Symp. on VLSI Technology, Systems and Applications, pp. 266-270, Taipei, Taiwan, R.O.C., May 1989.
- [106] W. Yang, "*Analog CCD Processors for Image Filtering*," SPIE Int'l. Symp. on Optical Eng. and Photonics in Aerospace Sensing, Orlando, FL, pp. 114-127, April 1991.
- [107] W. Yang and A.M. Chiang, "*VLSI Processor Architectures for Computer Vision*," Proc. of the Image Understanding Workshop, Palo Alto, CA, pp. 193-199, May 1989.
- [108] W. Yang, A.M. Chiang, "*A Full Fill-Factor CCD Imager with Integrated Signal Processors*," Proc. ISSCC, San Francisco, CA, pp. 218-219, February 1990.
- [109] C. Yarlagadda, "*A 512x512 Random Addressable Variable Resolution Image Sensor*," M.S. Thesis, Department of Electrical and Computer Engineering, New Jersey Institute of Technology, 1990.
- [110] P.C. Yu, S.J. Decker, H.S. Lee, C.G. Sodini and J.L. Wyatt, Jr., "*CMOS Resistive Fuses for Image Smoothing and Segmentation*," IEEE Journal of Solid-State Circuits, Vol. 27, No. 4, pp. 545-553, April 1992.
- [111] P.C. Yu and H.-S. Lee, "*A CMOS Resistive Fuse Processor for 2-D Image Acquisition, Smoothing and Segmentation*," submitted to 1992 European Solid-State Circuits Conference, Copenhagen, Denmark.

NSF/DARPA Workshop on Machine Learning and Vision: A Summary

Ryszard S. Michalski
Peter W. Pachowicz
Center for Artificial Intelligence
George Mason University

Azriel Rosenfeld
Yiannis Aloimonos
Center for Automation Research
University of Maryland

This report gives a brief account of the NSF/DARPA Workshop on Machine Learning and Vision, organized by George Mason University in collaboration with the University of Maryland, October 15-17, 1991 in Harpers Ferry, WV. The purpose of the workshop was to bring together researchers in vision and learning to discuss the possibilities of cross-fertilizing the two fields, and implementing learning capabilities in vision systems.

The workshop was attended by about 40 participants representing universities, industrial and governmental laboratories, and several sponsoring agencies. The workshop started with two general presentations, one on machine vision (A. Rosenfeld), and the second on machine learning (R.S. Michalski). Subsequent discussions were conducted in three sessions: 1) Learning in object recognition (organized by J. Shavlik and T. Poggio), 2) Learning in navigation (organized by T. Dean and T. Kanade), and 3) Learning in sensory-motor control (organized by R. Bajcsy and T. Mitchell).

The session on object recognition discussed issues related to types of tasks and important subtasks in object recognition. Two basic types were distinguished: learning shape descriptions and learning surface (texture) descriptions. The shape learning subtasks were classified according to their difficulty: isolated object recognition, recognition of specific objects in a scene, and recognition of objects that fulfill a functional goal (e.g., an object that could be used as a chair). The following issues were considered as important for learning in vision: relationship between 2D and 3D vision, number of training examples needed, use of prior knowledge, discovery of good representations, attribute selection, variability of the environment, and occlusion.

The session on learning in navigation classified the problems that require learning along such dimensions as: constrained vs. unconstrained navigation, static vs. dynamic navigation, and

structured vs. unstructured environment. Navigation tasks incorporating learning functions were analyzed according to shallow vs. deep inference, resource consideration, availability of supplementary knowledge, and complexity of behavior. Issues for machine learning include (i) learning in a constrained spatio-temporal context, (ii) building representations to facilitate planning in a spatio-temporal context, (iii) memory management during learning, (iv) combining sensor information, and (v) optimal feedback control.

The session on learning in sensory-motor control identified several bottleneck problems. The major goal of machine learning is viewed as automatic combining of specific vision and action modules in a task-independent way. This includes (i) learning efficient visual search, (ii) learning invariances that facilitate object identification under different imaging transformations and occlusions, (iii) learning module configuration and coordination for sensory-motor tasks, and (iv) learning calibration between sensing and action.

In summary, researchers agreed that many crucial elements of machine vision cannot be considered in isolation from machine learning. However, to be successful in the integration of learning and vision, researchers should pick a particular vision problem, apply acceptable restrictions on the problem, simplify the data, find solutions by applying learning technology, and then improve the solutions by gradually relaxing restrictions on the problem. It would be desirable to sponsor several long-term projects focused on both industrial and military applications. The creation of sharable testbeds is recommended for evaluation of results.

Reference

Machine Learning and Vision: Research Issues and Promising Directions, Report by Participants of the NSF/DARPA Workshop on Machine Learning and Vision (MLV-92), Harpers Ferry, WV, October 15-17, 1992, Edited by R. S. Michalski, A. Rosenfeld, P. W. Pachowicz and Y. Aloimonos, *Reports of Machine Learning and Inference Laboratory*, MLI 93-1, Center for Artificial Intelligence, George Mason University, Fairfax, VA, 1993.

The Workshop was sponsored jointly by the National Science Foundation and the Defense Advanced Research Projects Agency under Grant No. IRI-9208947.

DARPA Automatic Sensor Interpretation Workshop

Oscar Firschein

Software and Intelligent Systems Office, DARPA

March 4, 1993

Abstract

A DARPA workshop on automatic sensor interpretation was held to bring together government sponsors and people carrying out research and development in fields related to sensor interpretation. We indicate here some of the speakers and their topics. As a result of the Workshop, a BAA was issued for a university research initiative in theory and strategies for automatic target recognition (ATR).

1 Introduction

The DARPA Automatic Sensor Interpretation Workshop was held at George Mason University September 30 - October 2, 1992. The purpose was to bring together government sponsors and specialists in various disciplines related to sensor interpretation. The workshop was motivated by an automatic target recognition inter-office effort within DARPA involving the Advanced Systems Technology Office (ASTO), the Software and Intelligent Systems Technology Office (SISTO), The Defense Sciences office (DSO), and the Microtechnology Office (MTO).

As a result of the workshop, BAA 93-07, "University Research Initiative into the Theory and Strategies for Automatic Target Recognition," was issued on November 4, 1992. This BAA solicited proposals from U.S. graduate education institutions to research the theory and strategies for ATR. Of particular interest were those approaches that investigated the fundamental basis for detecting and recognizing dim or obscured, quasi-resolved targets located in severe ground clutter, and which would validate their results using real data. Evaluation of the proposals was completed in February 1993, and the contracts should be finalized by the middle of 1993.

2 Summary of the Workshop

The best way to capture the flavor of the workshop is to present the titles and authors of some of the key presentations.

2.1 Government Presentations

- "Energizing DARPA Application Programs with University Participation," Dr. Duane Adams, Deputy Director, DARPA
- "An Overview of the DARPA Surveillance and Targeting Programs," Dr. Larry Stotts, Asst. Director, Advanced Systems Technology Office, DARPA
- "Issues and Approaches in Automatic Target Recognition," Mr. Edward Zelnio, Air Force Wright Aeronautical Laboratory
- "Night Vision ATR Performance Evaluation," Dr. Clarence Walters, Night Vision and Electro-Optic Directorate
- "High Performance Sensor Processing," Dr. Dominick Giglio, Sensors and Processing, ASTO, DARPA
- "Image Understanding," Mr. Oscar Firschein, Image Understanding, Software and Intelligent Systems, DARPA
- "Spatial Reasoning for Tactical Fusion," Mr. Richard Anthony, Signals Warfare Directorate
- "Wavelet Techniques for Detection/Recognition," Lt. Col. Jim Crowley, Applied Science, Defense Sciences Office, DARPA
- "Neural and Gabor Techniques for ATR," Major Steve Rogers, Air Force Institute of Technology
- "Neural Nets for ATR," Dr. Barbara Yoon, Microtechnology Technology Office, DARPA

2.2 Technical Overviews

Some of the technical overviews were:

- "State of the Art Array Processing Algorithms for simultaneous Detection and Estimation," Prof. Harry Van Trees and Prof. Yariv Ephraim, George Mason University

- "Optimal Filtering for Multiband Detection," Prof. Irving Reed, University of Southern California
- "Image Processing and Analysis, an Overview," Prof. Azriel Rosenfeld, University of Maryland
- "Gabor, Wavelet, Morphological, and Distortion-Invariant Filter Sets," Prof. David Casasent, Carnegie Mellon University
- "The Challenges of Automatic Sensor Interpretation," Dr. Dan Dudgeon, MIT Lincoln Labs

2.3 Panel Discussions

The panel discussions were of particular interest:

- "Issues and Challenges in ATR," Prof. Clayton Stewart, George Mason University (moderator), Mr. Trent DePersia, ASTO-DARPA; Major Steve Rogers, Air Force Institute of Technology; Capt. Steve Suddarth, Air Force Office of Scientific Research, and Mr. Edward Zelnio, Air Force Wright Laboratory.
- "Issues and Challenges in Advanced Sensor Processing," Dr. Larry B. Stotts, ASTO-DARPA (moderator), Dr. Jurgen Gobien, SAIC, Dr. Les Novak, MIT Lincoln Labs; Dr. Serpil Ayashli, MIT Lincoln Labs; Dr. Dino Sofianos, SAIC, and Dr. Jack Ced-erquist, ERIM.

2.4 Technical Papers

The technical papers, too numerous to list, were in the areas of SAR, IR, and laser sensors; multisensor fusion, use of wavelets and Gabor techniques, neural networks, and image understanding. Of particular interest to the IU community were the following papers in the Image Understanding Session:

- "Computational Sensors," Prof. Takeo Kanade, Carnegie Mellon University
- "Active Vision, Task-oriented Vision," Prof. Chris Brown, University of Rochester
- "Understanding SAR images," Prof. Rama Chel-lapa, University of Maryland
- "Fusion of Multispectral and Panchromatic Imagery for Automated Cartography," Prof. Dave McKeown, Carnegie Mellon University

3 Conclusions

This workshop brought together people in various disciplines related to advanced ATR. As a result of the workshop it was realized that new ideas from the university community are required in ATR to deal with the new sophisticated threats such as fleetingly mobile targets in "deep hide," decoys and deception, and short exposure times. In addition, it is necessary for ATR systems to function in day or night, or adverse weather. It is hoped that the research contracts resulting from BAA 93-07, "University Research Initiative into the Theory and Strategies for Automatic Target Recognition," can provide some of these new ideas.

Section VII

Young Investigator Reports

Young Investigator Reports

Steven Abrams, Columbia University
Planning Viewpoints in an Active Environment

Bruce Draper, University of Massachusetts
Learning in Vision

Noah S. Friedland, University of Maryland
An Integrated Approach to Object Recognition

Robert Mandelbaum, University of Pennsylvania
Multi-Sensor Fusion for a Multi-Agent Robotic System

Ray Rimey, University of Rochester
Studying Control of Selective Perception with T-World and TEA

Jefferey Shufelt, Carnegie Mellon University
Incorporating Vanishing Point Geometry into a Building Extraction System

William M. Wells III, Massachusetts Institute of Technology
Statistical Object Recognition with the Expectation-Maximization Algorithm

Mourad Zerroug, University of Southern California
From Monocular Intensity Images to Volumetric Descriptions

Section VIII

Cartography/Photogrammetry Camera Calibration

Camera Calibration Using Line Correspondences

Richard Hartley

GE - Corporate Research and Development,
P.O. Box 8, Schenectady, NY, 12301.

Ph : (518)-387-7333

Fax : (518)-387-6845

email : hartley@crd.ge.com

Abstract

In this paper, a method of determining the essential matrix for uncalibrated cameras is given, based on line matches in three images. The three cameras may have different unknown calibrations, and the essential matrices corresponding to each of the three pairs of cameras may be determined. Determination of the essential matrix for uncalibrated cameras is important, forming the basis for many algorithms such as computation of invariants image rectification, camera calibration and scene reconstruction.

In the case where a fourth view is available, and all four cameras are assumed to have the same unknown calibration, the method of [Faugeras-Maybank-92a, Faugeras-Maybank-92b] may be used to calibrate the camera. The scene may then be reconstructed exactly (up to a scaled Euclidean transformation). This extends previous results of Weng, Huang and Ahuja ([Weng-92]) who gave a method for scene reconstruction from 13 line correspondences using a calibrated camera. The present paper shows that the camera may be calibrated at the same time that the scene geometry is determined.

1 Introduction

A traditional approach to analysis of perspective images in the field of Computer Vision has been to attempt to measure and model the camera that took the image. A large body of literature has grown up seeking to calibrate the camera and determine its parameters as a preliminary step to image understanding. The papers [Beyer-92] and [Beardsley-92] represent two of the latest approaches to camera calibration. A recent view ([Faugeras-92]) is that camera calibration is not desirable or necessary in many image understanding situations. Many authors have been led to consider uncalibrated cameras. The study of projective invariants ([Mundy-Zisserman-92]) is an example of a growing field based on the philosophy of avoiding camera calibration. In fact, study of uncalibrated cameras is intimately linked with the study of projective invariants, for a result of [Faugeras-92] and [Hartley-Gupta-92] shows that under most conditions a scene can be determined up to a projective transform of projective 3-space P^3 by a pair of images taken by uncalibrated cameras.

Central to the study of pairs of images is the essen-

tial matrix, introduced in [Higgins-81] for calibrated cameras, but easily extended to uncalibrated cameras. The essential matrix encodes the epipolar correspondences between two images. It has been shown to be a key tool in scene reconstruction from two uncalibrated views ([Faugeras-92, Hartley-Gupta-92]) as well as for the computation of invariants ([Hartley-93a]). The task of image rectification, which seeks to line up epipolar lines in a pair of images as a preliminary step to finding image correspondences, can be accomplished using the uncalibrated essential matrix ([Hartley-93c]) where previous methods have relied on camera modelling. It is particularly interesting that the essential matrix may be used for the calibration of a camera, and consequent scene reconstruction, given four or more views ([Faugeras-Maybank-92a, Faugeras-Maybank-92b]). This result provides a strong argument for not assuming camera calibration *a priori*, and underlines the central rôle of the essential matrix.

A recent paper Weng, Huang and Ahuja ([Weng-92]) gave an algorithm for reconstructing a scene from a set of at least 13 line correspondences in three images. They assumed a calibrated camera in their algorithm. It is the purpose of the present paper to extend their result to uncalibrated cameras, showing that the essential matrices can be computed from three uncalibrated views of a set of lines. It is not assumed that the three cameras all have the same calibration. In fact, the essential matrices corresponding to each of the three image pairs may be computed. In the case where four views with the same camera are available, however, the result of [Faugeras-Maybank-92a, Faugeras-Maybank-92b] may be applied to obtain the complete calibration of the four cameras and reconstruct the scene up to a scaled Euclidean transformation. Thus, in this case it is shown that the assumption of calibrated cameras is unnecessary, for the cameras may be calibrated at the same time that the scene is reconstructed.

One unfortunate aspect of the algorithm [Weng-92] is that 13 line correspondences in three images are necessary, compared with eight point correspondences (and with some effort only six, [Hartley-93b]). Although nothing can be done with two views or fewer (see [Weng-92]), a counting argument shows that as few as nine lines in three views may be sufficient, al-

though it is extremely unlikely that a linear or closed form algorithm can be found in this case. It is shown in section 4 of this paper that if four of the lines are known to lie in a plane, then a linear solution exists with only nine lines.

2 Preliminaries

2.1 Notation

Consider a set of points $\{x_i\}$ as seen in two images. The set of points $\{x_i\}$ will be visible at image locations $\{u_i\}$ and $\{u'_i\}$ in the two images. In normal circumstances, the correspondence $\{u_i\} \leftrightarrow \{u'_i\}$ will be known, but the location of the original points $\{x_i\}$ will be unknown. Normally, unprimed quantities will be used to denote data associated with the *first image*, whereas primed quantities will denote data associated with the *second image*.

Since all vectors are represented in homogeneous coordinates, their values may be multiplied by any arbitrary non-zero factor. The notation \approx is used to indicate equality of vectors or matrices up to multiplication by a scale factor.

Given a vector, $t = (t_x, t_y, t_z)^T$ it is convenient to introduce the skew-symmetric matrix

$$[t]_x = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} \quad (1)$$

This definition is motivated by the fact that for any vector v , we have $[t]_x v = t \times v$ and $v[t]_x = v \times t$.

The notation A^* represents the adjoint of a matrix A , that is, the matrix of cofactors. If A is an invertible matrix, then $A^* \approx (A^T)^{-1}$.

2.2 Camera Models

Nothing will be assumed about the calibration of the two cameras that create the two images. The camera model will be expressed in terms of a general projective transformation from three-dimensional real projective space, P^3 , known as object space, to the two-dimensional real projective space P^2 known as image space. The transformation may be expressed in homogeneous coordinates by a 3×4 matrix P known as a camera matrix and the correspondence between points in object space and image space is given by $u_i \approx P x_i$.

For convenience it will be assumed throughout this paper that the camera placements are not at infinity, that is, that the projections are not parallel projections. In this case, a camera matrix may be written in the form

$$P = (M \mid -Mt)$$

where M is a 3×3 non-singular matrix and t is a column vector $t = (t_x, t_y, t_z)^T$ representing the location of the camera in object space (see [Hartley-93a]).

2.3 The Essential Matrix

For sets of points viewed from two cameras, Longuet-Higgins [Higgins-81] introduced a matrix that has subsequently become known as the essential matrix. In Longuet-Higgins's treatment, the two cameras

were assumed to be calibrated, meaning that the internal camera parameters were known. It is not hard to show (for instance see [Hartley-92]) that most of the results also apply to uncalibrated cameras of the type considered in this paper.

The following basic theorem is proven in [Higgins-81].

Theorem 1. (Longuet-Higgins) *Given a set of image correspondences $\{u_i\} \leftrightarrow \{u'_i\}$ there exists a 3×3 real matrix Q such that*

$$u'_i{}^T Q u_i = 0$$

for all i .

The matrix Q is called the essential matrix. Next, we consider the question of determining the essential matrix given the two camera transformation matrices. The following result was proven in [Hartley-92].

Proposition 2. *The essential matrix corresponding to a pair of camera matrices $P = (M \mid -Mt)$ and $P' = (M' \mid -M't')$ is given by*

$$Q \approx M'^* M^T [M(t' - t)]_x.$$

For a proof of Proposition 2 see [Hartley-92].

2.4 Computing Lines in Space

Lines in the image plane are represented as 3-vectors. For instance, a vector $l = (l, m, n)^T$ represents the line in the plane given by the equation $lu + mv + nw = 0$. Similarly, planes in 3-dimensional space are represented in homogeneous coordinates as a 4-dimensional vector $\pi = (p, q, r, s)^T$.

The relationship between lines in the image space and the corresponding plane in object space is given by the following lemma.

Lemma 3. *Let λ be a line in P^3 and let the image of λ as taken by a camera with transformation matrix P be l . The locus of points in P^3 that are mapped onto the image line l is a plane, π , passing through the camera centre and containing the line λ . It is given by the formula $\pi = P^T l$.*

Proof. A point x lies on π if and only if it is mapped to a point on the line l by the action of the transformation matrix. This means that Mx lies on the line l , and so

$$l^T Mx = 0. \quad (2)$$

On the other hand, a point x lies on the plane π if and only if $\pi^T x = 0$. Comparing this with (2) lead to the conclusion that $\pi^T = l^T M$ or $\pi = M^T l$ as required. \square

2.5 Degrees of Freedom

In this section, we compute how many views of a set of lines are necessary to determine the positions of the lines in space. Suppose that n unknown lines are visible in k views with unknown camera matrices. Suppose that the images of the lines in each of the k

views are known. Each line in each view gives rise to two equations. In particular, suppose λ is a line in \mathcal{P}^3 and l is the image of that line as seen by a camera with camera matrix P . Let x be a point on λ , then as shown in (2) $l^T P x = 0$. Since the line λ can be specified by two points, two independent equations arise. The total number of equations is therefore equal to $2nk$.

On the other hand, each line in \mathcal{P}^3 has four degrees of freedom, so up to projectivity, n lines have a total of $4n - 15$ degrees of freedom, as long as $n \geq 5$.¹ Furthermore, each camera matrix has 11 degrees of freedom. In summary :

$$\begin{aligned} \# \text{ D.O.F} &= 4n - 15 + 11k, \\ \# \text{ equations} &= 2nk. \end{aligned}$$

To solve for the line locations,

$$2nk \geq 4n + 11k - 15. \quad (3)$$

In particular for 6 lines at least 9 views are necessary. On the other hand, for just 3 views, at least 9 lines are necessary.

Once the lines are known, the camera matrices may be computed using (2), and the essential matrices of each pair may be computed using Theorem 2.

The bounds given by (3) are minimum requirements for the computation of the essential matrices of all the views. The necessity for at least 9 lines in 3 views just demonstrated should be compared with section 3 in which a linear method is given for computing Q from 13 lines in 3 views. Also, compare with section 4 in which a linear method is given for computing Q under the assumption that four of the lines are coplanar.

3 Determination of the Essential Matrix from Line Correspondences

This section will investigate the computation of the essential matrix of an uncalibrated camera from a set of line correspondences in three views. As discussed in [Weng-92], no information whatever about camera placements may be derived from any number of line-to-line correspondences in two views. In [Weng-92] the motion and structure problem from line correspondences is considered. An assumption made in that paper is that the camera is calibrated, so that a pixel in each image corresponds to a uniquely specified ray in space relative to the location and placement of the camera. It will be shown in this section that this assumption is not necessary and that in fact the same approach can be adapted to apply to the computation of the essential matrix for uncalibrated cameras.

It will be assumed that three different views are taken of a set of fixed lines in space. That is, it is assumed that the cameras are moving and the lines are fixed, which is opposite to the assumption made in [Weng-92]. It will not even be assumed that the images are taken with the same camera. Thus the three cameras are uncalibrated and possibly different. The notation used in this section will be similar to that

used in [Weng-92]. Since we are now considering three cameras, the different cameras will be distinguished using subscripts rather than primes. Consequently, the three cameras will be represented by matrices

$$(M_0 | 0), (M_1 | -M_1 t_1) \text{ and } (M_2 | -M_2 t_2)$$

where t_1 and t_2 are the positions of the cameras with respect to the position of the zero-th camera, and M_i is a non-singular matrix for each i . For convenience, the coordinate system has been chosen so that the origin is at the position of the zero-th camera, and so $t_0 = 0$.

Now, consider a line in space passing through a point x and with direction given by a vector ℓ . Let N_i be the normal to the plane passing through the center of the i -th camera and the line. Then, N_i is given by the expression

$$N_i = (x - t_i) \times \ell = x \times \ell - t_i \times \ell.$$

Then for $i = 1, 2$,

$$\begin{aligned} N_0 \times N_i &= (x \times \ell) \times (x \times \ell - t_i \times \ell) \\ &= -\{(x \times \ell) \times (t_i \times \ell)\} \\ &= -\{((x \times \ell) \cdot \ell) t_i - ((x \times \ell) \cdot t_i) \ell\} \\ &= (N_0 \cdot t_i) \ell \end{aligned} \quad (4)$$

However, for $i = 1, 2$,

$$\begin{aligned} N_i \cdot t_i &= ((x - t_i) \times \ell) \cdot t_i \\ &= (x \times \ell) \cdot t_i - (t_i \times \ell) \cdot t_i \\ &= N_0 \cdot t_i \end{aligned}$$

Combined with the result of (4) this yields the expression

$$N_0 \times N_i = (N_i \cdot t_i) \ell \quad (5)$$

for $i = 1, 2$. From this it follows, as in [Weng-92] that

$$(N_2 \cdot t_2) N_0 \times N_1 = (N_1 \cdot t_1) N_0 \times N_2 \quad (6)$$

Now, let n_i be the representation in homogeneous coordinates of the image of the line ℓ in the i -th view. According to Lemma 3, N_i is the normal to the plane $(M_i | -M_i t_i)^T n_i$. Consequently,

$$N_i = M_i^T n_i.$$

Applying this to (6) lead to

$$\begin{aligned} (n_2^T M_2 t_2)(M_0^T n_0 \times M_1^T n_1) \\ = (n_1^T M_1 t_1)(M_0^T n_0 \times M_2^T n_2) \end{aligned} \quad (7)$$

We now state without proof a simple formula concerning cross products :

Lemma 4. If M is any 3×3 matrix, and u and v are column vectors, then

$$(Mu) \times (Mv) = M^*(u \times v). \quad (8)$$

¹As shown in [Hartley-93a] four lines have two degrees of freedom

Applying (8) to each of the two cross products in (7) leads to

$$\begin{aligned} M_0^{-1}(\mathbf{n}_2^T M_2 \mathbf{t}_2)(\mathbf{n}_0 \times M_0^* M_1^T \mathbf{n}_1) \\ = M_0^{-1}(\mathbf{n}_1^T M_1 \mathbf{t}_1)(\mathbf{n}_0 \times M_0^* M_2^T \mathbf{n}_2) . \end{aligned} \quad (9)$$

Now, cancelling M_0^{-1} from each side and combining the two cross products into one gives

$$\mathbf{n}_0 \times ((\mathbf{n}_2^T M_2 \mathbf{t}_2) M_0^* M_1^T \mathbf{n}_1 - (\mathbf{n}_1^T M_1 \mathbf{t}_1) M_0^* M_2^T \mathbf{n}_2) = 0 . \quad (10)$$

As in [Weng-92], we write

$$B = (\mathbf{n}_2^T M_2 \mathbf{t}_2) M_0^* M_1^T \mathbf{n}_1 - (\mathbf{n}_1^T M_1 \mathbf{t}_1) M_0^* M_2^T \mathbf{n}_2 \quad (11)$$

then $\mathbf{n}_0 \times B = 0$. Now, writing

$$\begin{aligned} M_0^* M_1^T &= \begin{pmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{pmatrix} \\ M_0^* M_2^T &= \begin{pmatrix} \mathbf{s}_1^T \\ \mathbf{s}_2^T \\ \mathbf{s}_3^T \end{pmatrix} \\ M_1 \mathbf{t}_1 &= \mathbf{t} \\ M_2 \mathbf{t}_2 &= \mathbf{u} \end{aligned} \quad (12)$$

vector B can be written in the form

$$B = \begin{pmatrix} \mathbf{n}_1^T (\mathbf{r}_1 \mathbf{u}^T - \mathbf{t} \mathbf{s}_1^T) \mathbf{n}_2 \\ \mathbf{n}_1^T (\mathbf{r}_2 \mathbf{u}^T - \mathbf{t} \mathbf{s}_2^T) \mathbf{n}_2 \\ \mathbf{n}_1^T (\mathbf{r}_3 \mathbf{u}^T - \mathbf{t} \mathbf{s}_3^T) \mathbf{n}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{n}_1^T E \mathbf{n}_2 \\ \mathbf{n}_1^T F \mathbf{n}_2 \\ \mathbf{n}_1^T G \mathbf{n}_2 \end{pmatrix} . \quad (13)$$

Where E , F and G are defined by this formula. Therefore, we have the basic equation

$$\mathbf{n}_0 \times \begin{pmatrix} \mathbf{n}_1^T E \mathbf{n}_2 \\ \mathbf{n}_1^T F \mathbf{n}_2 \\ \mathbf{n}_1^T G \mathbf{n}_2 \end{pmatrix} = 0 . \quad (14)$$

This is essentially the same as equation (2.13) in [Weng-92], derived here, however, for the case of uncalibrated cameras. As remarked in [Weng-92], for each line ℓ , equation (14) gives rise to two linear equations in the entries of E , F and G . Given 13 lines it is possible to solve for E , F and G , up to a common scale factor.

We now define a matrix Q_{01} by

$$Q_{01} = (\mathbf{t} \times \mathbf{r}_1, \mathbf{t} \times \mathbf{r}_2, \mathbf{t} \times \mathbf{r}_3)$$

This may be written as $Q_{01} = [\mathbf{t}]_{\times} (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$. Then, we see that

$$Q_{01}^T = - \begin{pmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{pmatrix} [\mathbf{t}]_{\times}$$

and in view of the definitions of \mathbf{r}_i and \mathbf{t} given in (12), we have

$$Q_{01}^T = M_0^* M_1^T [M_1 \mathbf{t}_1]_{\times}$$

from which it follows, using Proposition 2 that Q_{01} is the essential matrix corresponding to the (ordered) pair of transformation matrices $(M_0 \mid 0)$ and $(M_1 \mid -M_1 \mathbf{t}_1)$.

From the definition of $E = \mathbf{r}_1 \mathbf{u}^T - \mathbf{t} \mathbf{s}_1^T$ it follows that $E^T (\mathbf{t} \times \mathbf{r}_1) = 0$. If E has rank 2, then $(\mathbf{t} \times \mathbf{r}_1)$ can be determined up to an unknown scale factor. If the same way, if F and G have rank 2, then $(\mathbf{t} \times \mathbf{r}_i)$ can be similarly determined. Since these three vectors are the columns of the essential matrix Q_{01} , it means that Q_{01} can be determined up to individually scaling its columns. How to handle the case where E , F or G does not have rank 2 is discussed in [Weng-92].

Now, by interchanging the roles of the first and second cameras in this analysis, it is possible to determine the matrix Q_{10} up to individual scalings of its columns. However, since $Q_{01} = Q_{10}^T$ the matrix Q_{01} can be determined up to scale.

4 Computation from 9 lines

If the lines are known to satisfy certain geometric constraints, then it is possible to compute the essential matrix using fewer lines in three views. The general idea is that if the projective geometry of some plane in the image can be fixed, then the determination of the epipolar geometry is simplified. This observation was applied to the determination of Q from point correspondences in [Zisserman-Hartley-93]. Instead of considering the configuration of 9 lines of which four are coplanar, we consider four points in a plane and five lines not in the plane. From four lines in a plane it is easy to identify four points as the intersections of pairs of lines. Thus, let $\mathbf{x}_1, \dots, \mathbf{x}_4$ be four points lying in a plane π in \mathcal{P}^3 . Let the images of these points as seen in three images be $\mathbf{u}_i, \mathbf{u}'_i$ and \mathbf{u}''_i . We suppose for convenience that the images have been subjected to appropriate projective transforms so that $\mathbf{u}_i = \mathbf{u}'_i = \mathbf{u}''_i$ for all i . Then, a necessary and sufficient condition for any further point \mathbf{x} to lie in the plane π is that \mathbf{x} projects to the same point in all three images.

This observation may be viewed in a different way. We may assume that the image planes of the three images are all identical with the plane π itself, since by an appropriate choice of projective coordinates in each of the image planes, it may be ensured that the projective mapping from plane π to each of the image planes is the identity coordinate map. The projective mapping associated with each camera maps a point \mathbf{x} in space to the image point \mathbf{u} in which the line through \mathbf{x} and the camera centre pierces the image plane. Coordinates for \mathcal{P}^3 may be chosen so that the plane π is the plane at infinity and the first camera is placed at the point $(0, 0, 0, 1)^T$. Let the other two cameras be placed at the points $\begin{pmatrix} \mathbf{a} \\ 1 \end{pmatrix}$ and $\begin{pmatrix} \mathbf{b} \\ 1 \end{pmatrix}$. The three camera transformation matrices are then $P = (I \mid 0)$, $P' = (I \mid -\mathbf{a})$ and $P'' = (I \mid -\mathbf{b})$. If we can compute the vectors \mathbf{a} and \mathbf{b} , then the essential matrices can be computed using Theorem 2.

Now consider a line λ in \mathcal{P}^3 which does not lie in the image plane. Let the projections of λ with respect to the three cameras be ℓ, ℓ' and ℓ'' . Since λ does not

lie in the image plane, its three images will be distinct lines. However, lines ℓ , ℓ' and ℓ'' must all meet at a common point, namely the point at which λ meets the image plane.

Given ℓ , ℓ' and ℓ'' the line λ may be retrieved as the intersection of the three planes defined by each line and its corresponding camera centre. Each such plane may be computed explicitly. In particular from (2)

the three planes are equal to $P^T \ell = \begin{pmatrix} \ell \\ 0 \end{pmatrix}$, $P'^T \ell' = \begin{pmatrix} \ell' \\ \mathbf{a}^T \ell' \end{pmatrix}$, and $P''^T \ell'' = \begin{pmatrix} \ell'' \\ \mathbf{b}^T \ell'' \end{pmatrix}$. The fact that these three planes meet in a common line implies that the 4×3 matrix

$$A = \begin{pmatrix} \ell & \ell' & \ell'' \\ 0 & \ell'^T \mathbf{a} & \ell''^T \mathbf{b} \end{pmatrix}.$$

has rank 2. Hence, there must be a linear dependency between the columns of A .

As remarked above, the lines ℓ , ℓ' and ℓ'' are coincident, so there is a relationship $\alpha \ell + \beta \ell' + \gamma \ell'' = 0$. This gives a linear dependency between the first three rows of A . Since ℓ , ℓ' and ℓ'' are known, the weights α , β and γ may be computed explicitly. Since A has rank 2, this dependency must also apply to the last row as well which means that

$$\beta \ell'^T \mathbf{a} + \gamma \ell''^T \mathbf{b} = 0.$$

This is a single linear equation in the coordinates of the two vectors \mathbf{a} and \mathbf{b} . Given five such equations, arising from five lines not lying in the plane π , it is possible to solve for \mathbf{a} and \mathbf{b} up to an unknown (but insignificant) scale factor.

Summary of the algorithm The algorithm for determining the essential matrices from four coplanar points and five lines in three images is as follows. We start with coordinates \mathbf{u}_i , \mathbf{u}'_i and \mathbf{u}''_i , the images of the points in the three images and also ℓ , ℓ' and ℓ'' , the images of the lines. The steps of the algorithm are as follows.

1. Determine two-dimensional projective transformations represented by non-singular 3×3 matrices K' and K'' such that for each $i = 1, \dots, 4$ we have $\mathbf{u}_i = K' \mathbf{u}'_i = K'' \mathbf{u}''_i$.
2. Replace each line ℓ'_i by the transformed line $K'^T \ell'_i$, and each ℓ''_i by $K''^T \ell''_i$.
3. For each $i = 1, \dots, 5$ find coefficients α_i , β_i and γ_i such that $\alpha_i \ell_i + \beta_i \ell'_i + \gamma_i \ell''_i = 0$.
4. Solve the set of five linear equations $\beta_i \ell'_i{}^T \mathbf{a} + \gamma_i \ell''_i{}^T \mathbf{b} = 0$ to find the vectors \mathbf{a} and \mathbf{b} , up to an indeterminate scale.
5. The three essential matrices are $K'^T [\mathbf{a}]_\times$, $K''^T [\mathbf{b}]_\times$ and $K''^T [\mathbf{b} - \mathbf{a}]_\times K'$.

The above discussion was concerned with the case in which the plane π was defined by four points. Any other planar object which uniquely defines a projective basis for the plane may be used just as well, for example four coplanar lines (as already noted). This shows that four coplanar lines plus five lines not in the plane are sufficient (in 3 views) to determine the essential matrices.

5 Conclusion

The two algorithms given above can be used to determine the essential matrices for the purposes of invariant computation, scene reconstruction, image rectification or some other purpose.

Most interesting would be the case in which we have four views with the same camera. Then the cameras can be calibrated and the entire scene reconstructed up to scaled Euclidean transform from line correspondences in three views. In order to implement this method, an efficient implementation of the calibration algorithm of Faugeras and Maybank ([Faugeras-Maybank-92a, Faugeras-Maybank-92b]) would be required. At the present time, no such implementation is available, so the calibration method described in this paper also remains unimplemented. This paper, therefore represents a contribution to the theory of calibration and scene reconstruction. It seems likely, however, that an efficient implementation of the algorithms of this paper and [Faugeras-Maybank-92a, Faugeras-Maybank-92b] will become available in the future.

References

- [Beyer-92] H. A. Beyer, "Accurate Calibration of CCD Cameras," Proceedings of IEEE Computer Vision and Pattern Recognition Conference, CVPR-92, pp. 96-101, 1992.
- [Faugeras-92] Faugeras, O., "What can be seen in three dimensions with an uncalibrated stereo rig?" Proc. of ECCV-92, G. Sandini Ed., LNCS-Series Vol. 588, Springer-Verlag, 1992, pp. 563 - 578.
- [Faugeras-Maybank-92a] O. D. Faugeras, Q.-T Luong and S. J. Maybank, "Camera Self-Calibration: Theory and Experiments," Proc. of ECCV-92, G. Sandini Ed., LNCS-Series Vol. 588, Springer-Verlag, 1992, pp. 321 - 334.
- [Faugeras-Maybank-92b] O. D. Faugeras and S. J. Maybank, "A theory of Self-Calibration of a Moving Camera" International Journal of Computer Vision, 8:2, pp. 123-151, (1992).
- [Hartley-92] R. Hartley, "Estimation of Relative Camera Positions for Uncalibrated Cameras," Proc. of ECCV-92, G. Sandini Ed., LNCS-Series Vol. 588, Springer-Verlag, 1992, pp. 579 - 587.
- [Hartley-Gupta-92] R. Hartley, R. Gupta and Tom Chang, "Stereo from Uncalibrated Cameras" Proceedings of CVPR92.
- [Hartley-93a] R. Hartley, "Chirality Invariants," in these proceedings.
- [Hartley-93a] R. Hartley, "Invariants of Lines in Space," in these proceedings.
- [Hartley-93b] R. Hartley, "Invariants of Points Seen in Multiple Images," preprint submitted for publication.
- [Hartley-93c] R. Hartley, "Computing Matched-epipolar Projections" preprint submitted for publication.
- [Higgins-81] H.C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," Nature, Vol. 293, 10 Sept. 1981.
- [Mundy-Zisserman-92] J. L. Mundy and A. Zisserman (editors), "Geometric Invariance in Computer Vision," MIT Press, Cambridge Ma, 1992.
- [Weng-92] J. Weng, T.S. Huang, and N. Ahuja, "Motion and Structure from Line Correspondences: Closed-Form Solution, Uniqueness and Optimization", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 14, No. 3, March, 1992.
- [Zisserman-Hartley-93] A. Zisserman, R. Hartley, J. Mundy, P. Beardsley, "Projective Structure From Multiple Views", in preparation.
- [Beardsley-92] P. Beardsley, D. Murray and A. Zisserman, "Camera Calibration Using Multiple Images", Proc. of ECCV-92, G. Sandini Ed., LNCS-Series Vol. 588, Springer-Verlag, 1992, pp. 312 - 320.

2-D Images of 3-D Oriented Points

David W. Jacobs
NEC Research Institute
4 Independence Way,
Princeton, NJ 08540

Abstract

Recent work has shown that a number of vision problems become simpler when one models projection from 3-D to 2-D as a non-rigid linear transformation. This projection model has been used fruitfully to solve problems in motion understanding, search based object recognition, the indexing of models in a data base, and in alignment types of recognition. These results have been largely restricted, however, to models and scenes that consist only of 3-D points. In this paper we show what happens when one attempts to extend these results to the somewhat more complicated domain of oriented points. In particular, we show how to perform indexing with these features, how to derive structure from motion sequences, and how to determine the image of a model from a small number of correspondences. However, we show that all of these problems become fundamentally more complex with oriented point features. More space is required for indexing, more images are required to derive structure, and new views cannot be synthesized linearly from old views. Moreover, these are not just idiosyncracies of our approach, they are inherent properties of the problems that we address.¹

1 Introduction

Recent work has shown that a number of vision problems become simpler when one models projection from 3-D to 2-D as a non-rigid linear transformation. These results have been largely restricted, however, to models and scenes that consist only of 3-D points. In this paper we show what happens when one attempts to extend these results to the somewhat more complicated domain of oriented points. In particular, we show how

to perform indexing with these features, how to derive structure from motion sequences, and how to determine the image of a model from a small number of correspondences. However, we show that all of these problems become fundamentally more complex with oriented point features.

In this work, one assumes that a 3-D object is transformed by an arbitrary affine transformation, followed by a scaled orthographic projection. Applying an affine transformation to a set of 3-D points is equivalent to applying an arbitrary 3×3 matrix to the points, and then translating them. We will call this a 3-D to 2-D linear transformation. Standard models of projection assume that a 3-D object is displaced in a scene with a rigid, Euclidean motion, and then projected to a 2-D image, using either perspective or scaled orthographic projection. Since a 3-D affine transformation is a non-rigid generalization of 3-D rigid transformations, the linear projection model includes more standard projections as a subset.

We focus on three pieces of past work that dealt with the linear projection of 3-D point features into 2-D image features. First, Ullman and Basri[15] show that any novel 2-D view of a rigid 3-D structure is a linear combination of a small number of basis views of the structure. This means that one can fully represent 3-D structure implicitly, with a few 2-D views, and use these to predict the appearance of the full structure based on the location of a few of its points. Second, Jacobs[5] considers the problem of using an ordered group of 2-D image points to index into lookup tables where one represents groups of 3-D model points. This indexing step finds geometrically consistent matchings between the image and the model. It is shown that one can optimally perform this indexing by representing each model group with a pair of lines in two orthogonal index spaces, using an image group to compute a key into each index space and then intersecting the result of two table lookups. (Lamdan and Wolfson[9] had previously used linear projections to devise a more limited 3-D to 2-D indexing method). Third, Koenderink and van Doorn[7] show that given two 2-D views of a set of 3-D points, one could infer the 3-D affine structure of those points. That is, one learns the 3-D structure up to an arbitrary affine transformation, which is all that one can determine given our projection model (see Shashua[12] for related results). There has

¹Research described in this paper was conducted at the MIT Artificial Intelligence Laboratory. Support for this research was provided in part by the University Research Initiative under Office of Naval Research contract N00014-86-K-0685, and in part by the Advanced Research Projects Agency under Army contract DACA76-85-C-0010 and under Office of Naval Research contract N00014-85-K-0124.

been other work that capitalizes on the linear projection model, but which is less directly related to our current results. This includes Roberts[11] early method of determining object poses for recognition, and Cass' [3] and Breuel's [2] recent work in object recognition.

All the work mentioned above is restricted to scenes consisting of 3-D points. (We use "scenes" or "models" interchangeably to refer to 3-D configurations of features that might be viewed from arbitrary directions). In this paper, we consider scenes that also include 3-D orientations. By orientation, we mean that one or more 3-D vectors are associated with some or all of the 3-D points, where only the directions and not the magnitudes of the vectors are known. For example, if we form a model from distinguished points on a curve along with their tangent vectors, we get an orientation vector associated with each point. If we use the vertices of a polyhedra as our model, the direction of the lines that form the vertices associates two or three direction vectors with each point. Oriented points, then, are of practical significance. They also provide perhaps the simplest generalization of past work. A different generalization from point features is described by Basri and Ullman[1], who consider the images produced by solid objects with smooth surfaces.

We will show that when we use oriented points the problems of image construction by linear combination, indexing, and affine structure from motion all become fundamentally more difficult. We show that novel views of an object are not linear combinations of past views, except in a trivial sense, although we do show how new views can be reconstructed nonlinearly based on old views. We show that indexing cannot be done by representing model groups using a pair of 1-D lines. To perform indexing, we must represent each group of model features by a 2-D surface in an index space. We show how to build this representation, but our results prove that representing groups of oriented point features in an index table inherently requires much more space than is needed to represent groups of simple point features. And we show that correspondences between features in four views of oriented 3-D points are needed to determine their affine structure, whereas only two views were needed to derive the structure of simple point features. We also show how to derive the 3-D structure of oriented point features by solving a simple set of linear equations.

In addition to presenting these concrete results, a second goal of this paper is to demonstrate the value of approaching such problems by first characterizing the set of images that a model can produce. We begin by providing a simple analytic mapping from a 3-D model to a geometric structure that represents all the images that the model can produce, when it is viewed using all possible linear transformations. Given this mapping, we show how past results concerning simple point features can be rederived and extended.

2 Descriptions of a Model's Images

We take a geometric approach to the problem of representing a model's images. We describe models using manifolds in *image space*. For our purposes a manifold may be thought of as just a simple n -dimensional sur-

face in a higher dimensional space. An image space is just a particular way of representing an image. If we describe an image using some parameters then each parameter is a dimension of image space, and each set of values of these parameters is a point in image space corresponding to one or more images that are described by this set of parameters. For example, if our image consists of a set of n 2-D points, then we can describe each image by the cartesian coordinates of these points, $(x_1, y_1, x_2, y_2, \dots, x_n, y_n)$. These coordinates describe a $2n$ -dimensional image space. Suppose then that our models consist of sets of 3-D points. As we apply all possible transformations to these models, we produce a large set of images that correspond to a manifold in our $2n$ -dimensional image space. We will therefore talk about a group of model features producing or corresponding to a manifold in image space. We will generally assume that these groups are ordered, avoiding the problem of finding correspondences, although canonical orderings can in some cases be defined (see Clemens and Jacobs[4]). Representing the manifolds of a group of features is just a method of representing its potential images, and implicitly, of representing its 3-D structure.²

We can use this geometric approach to solve a number of problems in which we reason about the 3-D structures that are consistent with one or more 2-D images. In this view, the problem of determining which scenes are compatible with a set of images becomes the problem of determining which manifolds in image space could contain the points that correspond to these images. We begin by reviewing results presented in Jacobs[5] for models consisting of 3-D point features. This will provide an example of our approach, and present results on which we will build.

We first note that there are several equivalent ways of formulating our projection model. We have already described it as applying an arbitrary 3-D affine transformation to a 3-D model, followed by scaled orthographic projection. For point features, a 3-D affine transformation is modeled by applying an arbitrary 3×3 matrix to the points, then adding an arbitrary 3-D translation vector. This is the method used by Koenderink and van Doorn[7]. This formulation is equivalent to assuming that our 3-D scene is projected in parallel, in an arbitrary direction, onto a 2-D plane, if we then allow the resulting 2-D image to be transformed by an arbitrary 2-D affine transformation. A 2-D affine transformation of an image is equivalent to taking a new photograph of the image, assuming scaled orthographic projection. This formulation is used in [5], and we will use it here. Both projection methods are equivalent to applying an arbitrary 3×2 matrix to point features, and then arbitrarily translating the resulting 2-D points.

There are two parts to describing the images that models can produce, given these transformations. First, what is the image space? Second, what manifolds do models correspond to in this image space? If we think about image formation as resulting from parallel projec-

²It is possible to define image spaces in which models do not correspond to manifolds, but such representations seem far-fetched, and we will not consider them here.

tion followed by a 2-D affine transformation, then it is natural to represent images in a way that is invariant under 2-D affine transformation. This allows us to separate the information in the image that depends on the model from the information that depends solely on the viewing transformation. We therefore represent images of point features as follows. We use the first three points of the image to define an affine basis. That is, if we denote the image points: (q_1, q_2, \dots, q_n) , let:

$$o = q_1 \quad u = q_2 - q_1 \quad v = q_3 - q_1$$

Then we may fully describe the locations of the remaining points using affine coordinates derived with respect to this basis. For example, we describe q_4 with the parameters (α_4, β_4) , where:

$$q_4 = o + \alpha_4 u + \beta_4 v$$

It is important to what follows that the affine coordinates of a point are left unchanged by any affine transform. That is, for any 2x2 matrix, A , and any 2-D translation, t , if $q_4 = o + \alpha_4 u + \beta_4 v$ then $Aq_4 + t = (Ao + t) + \alpha_4 Au + \beta_4 Av$.

An image is fully described by the parameters: $(o, u, v, (\alpha_4, \beta_4), \dots, (\alpha_n, \beta_n))$. Due to the model of projection we use we may ignore the first three of these parameters. To see this, we note that, except in degenerate cases, there exists an affine transform that will map any three image points to any other three image points. Therefore if a scene can produce the image, $(o, u, v, (\alpha_4, \beta_4), \dots, (\alpha_n, \beta_n))$, it can also produce the image $(o', u', v', (\alpha_4, \beta_4), \dots, (\alpha_n, \beta_n))$ for any choice of (o', u', v') , by combining the affine transform that maps (o, u, v) to (o', u', v') with the affine transform that was part of the projection that produced the original image. Meanwhile, this affine transformation will not effect the remaining parameters of the image. Therefore, the parameters (o, u, v) provide no information about whether a scene could produce an image.

The remaining image parameters form what we will call an *affine space*. An image with n ordered points is mapped into a point in a $2(n-3)$ -dimensional affine space by finding the affine coordinates of the image points, using the first three as a basis. We divide the affine space into two orthogonal subspaces, an α -space, and a β -space. The α -space is the set of α coordinates of the image's affine coordinates, and the β -space is similarly defined. The affine space is then equal to the cross product of the α -space and the β -space, and each image corresponds to a point in each of these two spaces. The previous paragraph states that the images that a scene can produce are fully described by the locus of points these images map to in affine space.

We now extend this image space to provide an affine invariant representation of oriented point features. To simplify this representation, we assume that each model contains at least three oriented points. We then continue to use three image points to define an affine basis, and describe the points' orientation vectors using this basis. Our image consists of points with associated direction vectors. Without loss of generality we may locate these vectors at the origin (see figure 1). We describe any additional image points using their affine coordinates, and

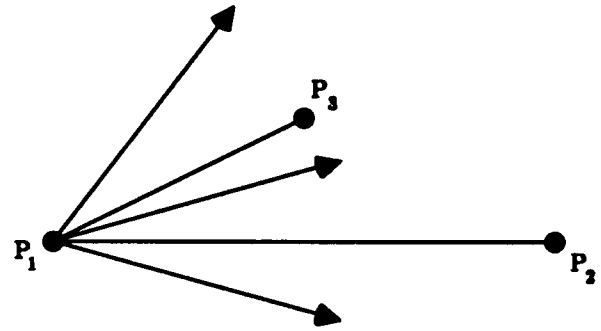


Figure 1: The three points shown are used as an affine basis, and the slopes of the vectors are found in this coordinate system.

we describe each orientation vector by its *affine slope*. The affine slope of a vector at the origin is just $\frac{\alpha}{\beta}$, where (α, β) is any point in the direction of the vector.

It is easily seen from the properties of affine transformations that the affine slope of a vector is well defined and is invariant under affine transformations. This representation of vectors is equivalent to an affine invariant representation derived by Van Gool et al.[16] using different methods. We use affine slope to define a new image space that combines point and vector information. We describe an image with the affine coordinates of any points beyond the first three, $(\alpha_4, \beta_4, \dots, \alpha_n, \beta_n)$, and with the affine slopes of all vectors, which we will call $(\theta_0, \dots, \theta_m)$. We call the space defined by these parameters *affine slope space*. As before, the problem of determining a scene's manifold becomes one of determining the set of affine invariant values it may produce when viewed from all directions. We may again ignore the actual position of the first three image points, except in using them to determine the affine invariant values of the remaining points and orientation vectors.

We have defined a simple image space for images of point features, and a slightly more complex space for oriented points. We now describe a mapping from any 3-D scene to its corresponding geometric shape in these image spaces. First, we assume that scenes of simple point features contain at least five points: p_1, p_2, p_3, p_4, p_j . Then we can show that the set of all affine coordinates that such a model can produce is described by a series of equations:

$$(\alpha_j, \beta_j) = (a_j, b_j) + \frac{((\alpha_4, \beta_4) - (a_4, b_4))}{r_j} \quad (1)$$

We have one such equation for each model point beyond the first four. The values of r_j, a_j and b_j are just measurable properties of the 3-D scene that do not depend on viewpoint at all. r_j is the ratio of the height of p_j above the plane formed by the first three scene points to the height of p_4 above this plane. And (a_j, b_j) are the affine coordinates of the projection of p_j down into this plane, using the first three scene points as an affine basis. Jacobs[5] derives equation 1.

This set of equations describes all images that the

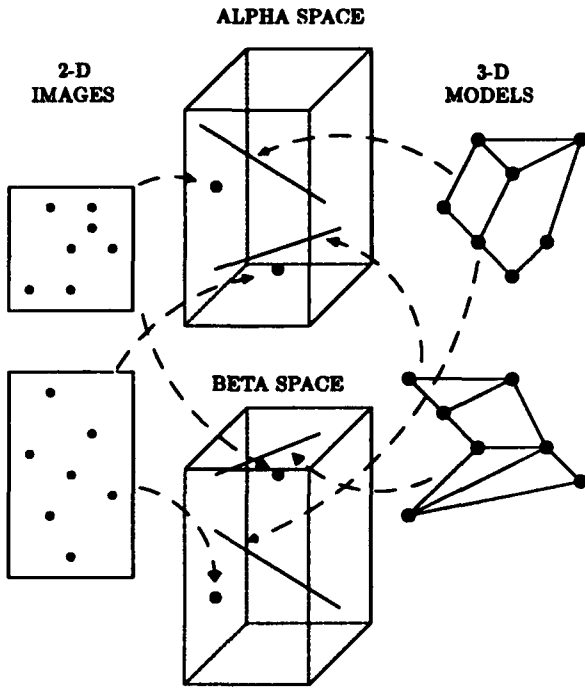


Figure 2: We may represent all of a model's possible images with a pair of lines in two, orthogonal spaces. Each image corresponds to a pair of points, one in each of these spaces.

scene points may produce. For any image, this equation will hold. And for any values described by the equation, there is a corresponding image that the scene may produce. There is one important degenerate case, in which these equations do not hold. If the scene points are coplanar, then their affine coordinates are invariant. In that case the above equation becomes degenerate; such a scene can produce only one set of affine coordinates.

Interpreted geometrically, the set of equations (1) describe a 2-D plane in the affine image space, since r_j , a_j and b_j are constant for a specific scene. This provides a concrete example of our overall approach. We develop a space such that all images correspond to points in that space and all scenes correspond to 2-D planes, or in the degenerate case to points, in that space. The problem of matching groups of features in one or more images to groups of features in possible scenes becomes the problem of determining which planes pass through one or more points in this space.

Taking the α and β components of these equations

$$\alpha_j = a_j + \frac{(\alpha_4 - a_4)}{r_j} \quad \beta_j = b_j + \frac{(\beta_4 - b_4)}{r_j}$$

we have equations that describe a line in α -space. We may derive a similar set of equations in β -space. These equations are independent. That is, for any set of α coordinates that a scene may produce in an image, it may still produce any feasible set of β coordinates. Notice that for any line in α -space, there is some scene whose images are described by that line. It is not true

that there is a scene corresponding to any pair of lines in α -space and β -space because the parameters r_j are the same in the equations for the two lines. This means that the two lines are constrained to have the same directional vector, but they are not further constrained. So we have further simplified our representation to one in which models correspond to 1-D lines and images to pairs of points. This is shown schematically in figure 2.

Now we determine the manifolds that correspond to scenes of oriented point features in affine slope space. We begin by introducing some special 3-D points related to our scene. With every orientation vector, v_i , we associate some point, r_i that is in the direction v_i from the origin. We denote the points of the scene by p_i . We may then describe the images that would be produced by the points $(p_1, p_2, p_3, r_0, \dots, r_m, p_4, \dots, p_n)$ with two lines in α and β space, which we call α^* and β^* . However, in practice we cannot know the images of the points r_i , only the direction of the vectors to them. We proceed by determining the images that are produced by these points, and then determining the affine slope of the vector to a point when only the direction of this vector is known. The vectors that we can extract from the image will be in the direction towards the location where the images of these points would be, however. This tells us that the affine slope of v_i 's image, called θ_i , will equal the α coordinate of r_i divided by the β coordinate of r_i . That is, $\theta_i = \frac{\alpha_{i+4}}{\beta_{i+4}}$. So for any two points on α^* and β^* , with coordinates $(\alpha_4, \alpha_5, \dots, \alpha_{n+m+1})$ and $(\beta_4, \beta_5, \dots, \beta_{n+m+1})$, the scene can produce an image which is described by the affine invariant parameters: $\frac{\alpha_4}{\beta_4}, \frac{\alpha_5}{\beta_5}, \dots, \frac{\alpha_{n+m+1}}{\beta_{n+m+1}}, \alpha_{m+5}, \dots, \alpha_{n+m+1}, \beta_{m+5}, \dots, \beta_{n+m+1}$. This is the mixture of affine coordinates and affine slopes that we call affine slope space.

We will now derive a set of equations that describe a scene's manifold in this space. We begin by showing how the possible values for θ_i that a scene produces can be expressed as a function of θ_0 , θ_1 , and characteristics of the scene. First some notation. We can describe α^* with a parameterized equation of the form:

$$\alpha^* = a + kv^*$$

where a is any point in α space on the line α^* , and has coordinates that we denote by $(a_0, a_1, \dots, a_{m+n-2})$, and v^* is a vector in α space that expresses the direction of α^* , and has coordinates $(v_0^*, \dots, v_{m+n-2}^*)$, and k is a variable. As k varies, we get the points on the line α^* . Similarly, we let:

$$\beta^* = b + cv^*$$

Note that v^* is the same in both equations, because the two lines must have the same directional vector, as we mentioned earlier.

We want to find the range of values for $(\theta_0, \dots, \theta_m)$, where, for a particular choice of k and c ,

$$\theta_i = \frac{\alpha_{i+4}}{\beta_{i+4}} = \frac{a_i + kv_i^*}{b_i + cv_i^*}$$

That is, any possible set of affine slopes the scene may produce is found by finding a set of α values that the

constructed r_i points can produce, and then dividing these by a set of β values that could be produced. This equation expresses these possible θ values as a function of the parameters k and c , which may vary arbitrarily.

We ignore the degenerate cases where α_4 and β_4 are constant over α^* or β^* . These are the cases in which the first scene vector is coplanar with the three scene points, and so its affine slope does not vary with viewing direction. Then we may choose for a and b those points on α^* and β^* for which $a_0 = 0$ and $b_0 = 0$. This gives us the equation:

$$\theta_0 = \frac{a_0 + kv_0^*}{b_0 + cv_0^*} = \frac{k}{c}$$

This implies

$$c = \frac{k}{\theta_0}$$

We can use this to get:

$$\begin{aligned}\theta_1 &= \frac{a_1 + kv_1^*}{b_1 + \frac{kv_1^*}{\theta_0}} \\ \theta_1(b_1\theta_0 + kv_1^*) &= a_1\theta_0 + kv_1^*\theta_0 \\ k(v_1^*\theta_1 - v_1^*\theta_0) &= a_1\theta_0 - b_1\theta_0\theta_1 \\ k &= \frac{\theta_0(a_1 - \theta_1b_1)}{v_1^*(\theta_1 - \theta_0)}\end{aligned}$$

So we can express k and c in terms of the first two affine slopes that we detect in the image and of properties of the scene that determine the lines α^* and β^* . Implicitly, we have used θ_0 and θ_1 to solve for the viewpoint. This allows us to express each remaining image parameter, both affine slopes and the α and β coordinates that describe other image points, as a function of these first two affine slopes and properties of the scene. We find:

$$\begin{aligned}\alpha_{j+4} &= a_j + v_j^*k \\ &= a_j + \frac{v_j^*\theta_0(a_1 - \theta_1b_1)}{v_1^*(\theta_1 - \theta_0)} \\ \beta_{j+4} &= b_j + v_j^*c \\ &= b_j + \frac{v_j^*(a_1 - \theta_1b_1)}{v_1^*(\theta_1 - \theta_0)} \\ \theta_i &= \frac{a_i + v_i^*k}{b_i + v_i^*c} \\ &= \frac{a_i + \frac{v_i^*\theta_0(a_1 - \theta_1b_1)}{v_1^*(\theta_1 - \theta_0)}}{b_i + \frac{v_i^*(a_1 - \theta_1b_1)}{v_1^*(\theta_1 - \theta_0)}} \\ &= \frac{a_i v_1^*(\theta_1 - \theta_0) + v_i^*\theta_0(a_1 - \theta_1b_1)}{b_i v_1^*(\theta_1 - \theta_0) + v_i^*(a_1 - \theta_1b_1)} \\ &= \frac{-b_1 v_1^*\theta_0\theta_1 + (a_1 v_1^* - a_i v_1^*)\theta_0 + a_i v_1^*\theta_1}{-b_i v_1^*\theta_0 + (b_i v_1^* - b_1 v_1^*)\theta_1 + a_1 v_1^*}\end{aligned}$$

Note that this derivation fails in a few degenerate cases. In particular, it will fail if any of the scene points or vectors is coplanar with the first three scene points, in which case that affine value is constant. A physically unrealizable solution to the above equation occurs whenever $\theta_0 = \theta_1 = \theta_i$, in which case our derivation involved

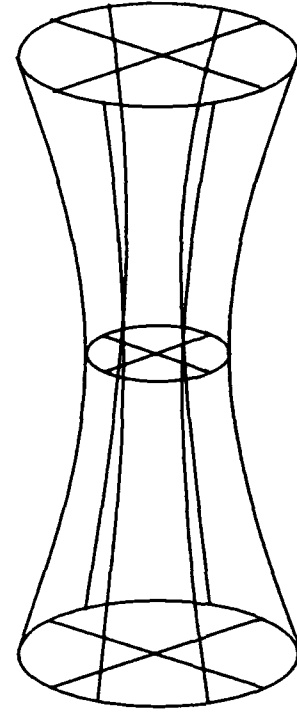


Figure 3: A hyperboloid of one sheet.

dividing by 0. This reflects the fact that as our viewpoint becomes closer to the plane of the first three scene points, the affine slopes all converge to the same value. They never quite get there, since if we view the scene from a point in this plane, the first three image points will be collinear, and all the affine coordinates become undefined. As our viewpoint approaches this plane and rotates about it, all the affine slopes can approach any common value.

So we have an analytic form describing all images of the model. We now show in the case of 3 points and 3 vectors that this form describes a 2-D hyperboloid in a 3-D image space.

We introduce the following abbreviations:

$$\begin{aligned}c_1 &= a_1 v_2^* & c_2 &= a_2 v_1^* & c_3 &= b_1 v_2^* & c_4 &= b_2 v_1^* \\ x &= \theta_0 & y &= \theta_1 & z &= \theta_2\end{aligned}$$

We note that c_1, c_2, c_3, c_4 are properties of the scene, and that scenes may be chosen to produce any set of these values. So, the set of manifolds that can be produced is precisely described by:

$$z = \frac{-c_3 xy + (c_1 - c_2)x + c_2 y}{-c_4 x + (c_4 - c_3)y + c_1}$$

$$-c_4 xz + (c_4 - c_3)yz + c_1 z + c_3 xy - (c_1 - c_2)x - c_2 y = 0 \quad (2)$$

Adopting the notation of Korn and Korn (pp. 74-76)[8] we find:

$$I = 0 \quad D = -2c_3 c_4 (c_4 - c_3) \quad A = (c_1 c_4 - c_2 c_3)^2 \geq 0$$

This tells us that when we look at three dimensions of affine slope space, we find that a model's manifold is a hyperboloid of one sheet (see figure 3). That is, the set

of affine slopes that three oriented points produce form an hyperboloid in the space of possible affine slopes. We also see that we can find a model corresponding to any hyperboloid that fits equation 2. For our purposes, we do not need to consider the degenerate cases in detail.

To recap this section, we have shown how to determine the manifolds in image space corresponding to scenes of both plain and oriented point features. This allows us to consider the problem of matching models and images by considering an equivalent problem of matching points in image space to lines or hyperbolic shapes. We now show how to use this formulation of matching to solve real problems.

3 Affine Structure from Motion

Koenderink and van Doorn[7] and Shashua[12] have noted that two views of an object can be used to predict additional views, and have applied this result to motion understanding. Koenderink and van Doorn show that the *affine structure* of an object made of 3-D points can be derived from two views. Affine structure is that part of the object's geometry that remains unchanged when we apply an arbitrary affine transformation to the object's points. For example, given two views of five corresponding points, they compute a 3-D affine invariant representation of the fifth point with respect to the first four. This is similar to the affine coordinates that we have used above, which are an affine invariant description of a 2-D object. Then, given the location of the first four points in a third image, the location of the fifth point may be determined. This result is particularly significant because it is known (Ullman[14]) that three views of an object are needed to determine the object's rigid structure when images are formed with scaled orthographic projection.

We can rederive this theorem from our previous results. Determining the affine structure of a model is equivalent to determining the manifold that it corresponds to in image space. Recall that the projection model that we use is equivalent to applying a 3-D affine transformation to a model, followed by orthographic projection. This means that if two objects have the same 3-D affine structure, then, and only then, can they produce exactly the same set of images under our projection model. There is a one-to-one mapping between affine structure and a model's manifold in an affine-invariant image space.

We have shown that a 3-D scene of points corresponds to a pair of lines in α and β space that have the same direction. Given two images of a model seen from different directions, we can determine two sets of affine coordinates that the model may produce. That is, we have two points in α space and two points in β space that must be included in the model's manifold. We can use these pairs of points to determine the two lines that must correspond to the model in α and β space. In fact, since the two lines have the same directional vector, it is sufficient to know the affine coordinates of one image of a scene, and just the α or the β values of a second image to determine the manifold of the scene that produced the two images. This implicitly tells us the affine structure

of the scene, and everything that we could know about which images it can produce.

But we may go beyond this, and consider what happens when we try to extend Koenderink and van Doorn's result to oriented points features. We find that four views are needed to determine the affine structure of oriented points. We consider a model with three points and three orientation vectors. We know that for any hyperboloid of the following form:

$$-c_4xz + (c_4 - c_3)yz + c_1z + c_3xy - (c_1 - c_2)x - c_2y = 0$$

there is a model whose images are described by this hyperboloid, where x, y, z are the affine slopes of the three image vectors, and c_1, c_2, c_3, c_4 are parameters of the model, which may take on any values. Determining the affine structure of the model is equivalent to finding the values of c_1, c_2, c_3, c_4 . If we do not know these values, we do not know the set of images that the model can produce and so we can not know the model's affine structure.

Every image of the model gives us a set of values for x, y , and z , while c_1, c_2, c_3 , and c_4 remain to be determined. This gives us one linear equation in four variables. We need four independent equations to solve for these variables, and hence we need at least four views of the object to find its affine structure. Given three views of the scene, there will still be an infinite number of different hyperboloids that might produce those three images, but that would each go on to produce a different set of images.

This result is easily extended to four or more oriented points. The affine structure of additional points can be found using the method for point features described above. Additional orientation vectors each provide four new unknowns, and one new equation for each image.

However, it only takes three views to determine the rigid structure of four or more oriented points. To compute this, we can first use the locations of the points in three views to determine their 3-D location, as shown by Ullman. This tells us the 3-D location of each oriented point and each viewing direction, but not the 3-D direction of the orientation vectors. A view of an orientation vector at a known 3-D location restricts that vector to lie in a plane. For each orientation vector, two views tell us two different planes that include the vector. As long as our viewpoints are not identical, these planes intersect in a line, which tells us the direction of the orientation vector.

It might seem paradoxical that from three views we can determine the rigid structure of oriented points, while we need four views to determine their affine structure. But keep in mind that a view of an object provides us with less information about the object if we assume the view was created with a linear transformation than if we assume scaled orthographic projection.

This result shows us how to solve for the affine structure of an object of oriented points by just solving a set of linear equations, provided that we have four views of the object. But the need for four views is a significant limitation. Koenderink and van Doorn suggested that affine structure is an intermediate representation that

we can compute using less information than is required to determine rigid structure. However, we see that this is not always true.

4 Indexing

Indexing is a general term often used to describe any object recognition process that selects a small number of candidate objects, or object parts, from a large data base of different possibilities on the basis of information available in the image. In one approach to indexing, we preprocess the models, making entries in a lookup table that each point to a portion of a model. Then at run time, we use some set of features in the image to compute an index into the lookup table. Indexing the table at that point, we find all portions of all known models that could have produced that particular group of image features. The indexing process enforces geometric consistency between image features and the model features to which we match them.

The central problem of indexing with a single table lookup is to find the best way of representing models as manifolds in image space. For if we look in an index table just once for an ordered group of image features, this means that the index table is an image space; each point in the table corresponds to one or more images. If we are to avoid missing correct matches, then for each group of model features we must make entries in the table at every point that corresponds to a possible image of those model features. This means that the table represents each group of model features' manifold in image space in some discrete form.

So to perform indexing we would like to have an analytically understood mapping from the set of possible groups of model features to their manifolds in some image space. Then we may divide image space into discrete buckets, and determine which buckets are intersected by each model group's manifold. We then place pointers to the model group in each of these buckets. The amount of space required by this approach is approximately Nd^n , where we represent N model groups in the lookup table, we divide each dimension of the table into d parts, and where each manifold is n -dimensional. This space can be considerable, and therefore it is important to find an indexing method that uses the lowest-dimensional manifolds possible.

Our past results show that we can represent a model of point features using two 1-D lines in two orthogonal image spaces. This tells us that we can perform indexing using α and β spaces that we have divided into discrete buckets. For each model group, we then make entries in those buckets intersected by the line in each space that corresponds to that model group. Then at run time, we compute the α and β values that describe an image, and use them to index separately into the two lookup tables, intersecting the results. The space required for this grows at only a linear rate as we discretize the spaces more finely. This indexing system is implemented and described in [5]. There we also show, based on results in Clemens and Jacobs[4] that this is the most space efficient possible way of representing groups of 3-D model points in a lookup table.

In this paper we have shown an analytic mapping from groups of oriented point features to manifolds in an affine slope space. These manifolds are all 2-D. We could therefore perform indexing by discretely representing affine slope space. We can see that by characterizing the geometric structures that represent the images that a model can produce we are solving the primary theoretical problem that arises in indexing. It can require a good deal of space, however, to discretely represent a 2-D manifold in a lookup table. For example, Thompson and Mundy[13], Lamdan and Wolfson[9], and Clemens and Jacobs[4] all build such lookup tables by uniformly sampling the set of images that a model group can produce, instead of analytically computing these images. Thompson and Mundy, in fact, do this for groups consisting of pairs of vertices. In all three systems thousands of table entries are needed to represent each group of model features. We might want to represent many different objects in a lookup table, and a model of each object may give rise to many different subgroups of features that we would want to individually recognize. And in these systems, the lookup tables are discretized fairly coarsely. A more fine discretization might be desirable, but this would require even more space. So it is of considerable practical significance to determine whether we can represent model groups with lower-dimensional manifolds.

Unfortunately, we can prove that this is not possible. First we note that if we represent oriented point features with a single manifold in a single image space, these manifolds must be at least 2-D. The proof of this is given in Jacobs[6], and is essentially the same as the proof given for simple point features in Clemens and Jacobs[4]. Now in the case of point features we were able to decompose a single affine space into two orthogonal subspaces such that each model was represented with two 1-D manifolds. Our only hope of reducing the space required to index oriented point features is that we can similarly decompose the image space that represents oriented point features so that all the 2-D manifolds are decomposed into pairs of 1-D manifolds. We show this cannot be done.

Our proof will assume that each model contains at least three points, and three or more vectors. We assume that any configuration of points and vectors is a possible model group. We also assume a continuous mapping from images to our image space, and to any image subspaces. This is essential in any practical indexing scheme, because otherwise a small perturbation in an image, due to error, could result in large changes in that image's representation in image space. We show that if there is a decomposition of the image space that decomposes all the manifolds in it, then the kinds of intersections that can occur between manifolds must be limited, and that the class of manifolds produced by oriented point models do not meet these restrictions.

We will suppose the opposite of our proposition, that image space may be decomposed into two subspaces, such that each 2-D manifold in affine slope space that corresponds to a model is the cross-product of two 1-D curves in each of the subspaces. Then when two manifolds intersect in image space, we can determine the

places where they intersect by taking the cross product of the intersections of their 1-D manifolds in the two image subspaces. Suppose that two models' manifolds intersect in image space in a 1-D curve. Then our decomposition of image space must represent this curve as the cross product of a 1-D curve in one image space, and a point in the second image space. This means that in one of the two image subspaces, the two 1-D curves that represent the two models must overlap, so that their intersection is also a curve and not just a point.

This observation allows us to formulate a plan for deriving a contradiction. We pick a model group, M , with manifold H . We then choose a point P on H (that is, P corresponds to an image of M). We define p and p' as the points that correspond to P in the first and second image subspaces respectively. We will construct five new, special models, M_1, M_2, M_3, M_4, M_5 . Each of these model's manifolds will intersect H in a 1-D curve in image space. We call these curves K_1, K_2, K_3, K_4, K_5 . Each of these curves will contain P , by construction. Then, since each curve maps to a curve in one image subspace, and a point in the other, we may assume without loss of generality that K_1, K_2 , and K_3 map to the curves k_1, k_2 , and k_3 in the first image subspace, and to the points r_1, r_2 and r_3 in the second image subspace. Then, in order for the curves K_1, K_2 , and K_3 to all include the point P , it must be that $r_1 = r_2 = r_3 = p'$, and that k_1, k_2 , and k_3 all intersect at the point p in the first image subspace. We will call the curve that represents M in the first image subspace k . k_1, k_2 , and k_3 must all lie on k because they come from the intersection of M and other models. It is possible that two of these curves intersect only at p if they end at p , and they occupy portions of k on opposite sides of p . But with three curves, two at least (suppose they are k_1 and k_2) must intersect over some 1-D portion of k . Since they both intersect at p' in the other image space, this will tell us that K_1 and K_2 intersect over some 1-D portion of image space. We will then derive a contradiction by showing that in fact all of the curves, K_1, K_2, K_3, K_4, K_5 , intersect each other only at a single point, P . So, to summarize the steps needed to complete this proof, we will: construct the point P and the models $M, M_1, M_2, M_3, M_4, M_5$ so that each additional model's manifold intersects M 's in a 1-D curve that includes P . We will then show that these curves intersect each other only at P , that is, that M and any two of the other models have only one common image.

For these constructions, we will choose our models to be identical and planar, except for their first three orientation vectors. Therefore, in considering the intersection of these models' manifolds, we need only consider their intersection in the coordinates $(\theta_0, \theta_1, \theta_2)$, since their remaining coordinates will always be constant, and will be the same for each model. Therefore, when we speak of the coordinates of a point in image space, we will only consider these three coordinates. And to describe the values for $(\theta_0, \theta_1, \theta_2)$ that a model can produce, we need only give the values for c_1, c_2, c_3, c_4 that will describe the model's hyperboloid in $(\theta_0, \theta_1, \theta_2)$ space.

It is easy to see from equation 2 that, in general, any two of these hyperboloids will intersect in a set of 1-D

surfaces, and any three hyperboloids will intersect only at points, and in the line $\theta_0 = \theta_1 = \theta_2$, as noted above. Therefore, any general set of six hyperboloids chosen to intersect at a common point will fulfill our needed construction.

We can also prove this result another way, which will perhaps strengthen the reader's intuitions about these hyperboloids. Let H be a 3-D hyperboloid, and P be an arbitrary point on it. We derive a contradiction after assuming that we can decompose H into two 1-D curves in two image spaces. Suppose that P is represented again by the two points p and p' in the two image spaces. Choose any other point Q on H . Referring to equation 2 we see that knowing two points of a hyperboloid gives us two linear equations in the four unknowns that describe the hyperboloid. Therefore, we may readily find a second hyperboloid, H' , that also includes P and Q , but that does not coincide with H . As noted above, in general H and H' intersect in a 1-D curve, which must correspond to a curve in one image space, and to either p or p' in the other. In particular, this means that Q must correspond to either p or p' . Since Q is an arbitrary point, all points on H must correspond to either p or p' . This contradicts our assumption that H is represented by the cross-product of two curves.

Notice that these proofs do not depend on our choice of affine slope space to represent images of oriented points. These proofs make use only of the topology of the intersection of manifolds. This topology will be preserved by any one-to-one continuous mapping, and hence will be present in any continuous representation of images.

We have therefore shown that our representation of groups of oriented points as 2-D manifolds is optimal in terms of its dimensionality. We cannot represent images of such models using lower-dimensional manifolds. This places an unexpected lower bound on the cost of indexing groups of oriented point features. It shows that it requires considerably more space to index them than to index simple point features. Again we see that adding orientations to our models makes a basic vision task fundamentally more difficult.

5 Recognition by Linear Combinations

Ullman and Basri[15] show that any image of a model of 3-D points can be expressed as a linear combination of a small set of basis images of the object. That is, given a few views of an object, $i_1 \dots i_n$, and any new view, i_j , we can find coefficients $a_1 \dots a_n$ so that:

$$i_j = \sum_{k=1}^n a_k i_k$$

where we multiply and sum images by just multiplying and summing the cartesian coordinates of each image point separately. This idea is refined independently by Basri and by Poggio[10] into the following form.

Suppose we have a model, m , with n 3-D points. i_1 and i_2 are two images of m . We describe each image with cartesian coordinates, and assume there is no translation in the projection. Let x_1 be an n -dimensional vector

containing all of i_1 's x coordinates, and let y_1 be an n -dimensional vector of its y coordinates. Similarly, define x_2 and y_2 for i_2 . Take any new image of m , i_j , and define x_j and y_j . Then Basri and Poggio show that there exist a_0, a_1, a_2 and b_0, b_1, b_2 such that:

$$x_j = a_0 x_1 + a_1 y_1 + a_2 x_2$$

$$y_j = b_0 x_1 + b_1 y_1 + b_2 x_2$$

This tells us that the x and y coordinates of a new image are a linear combination of one and a half views of the object, that is, of the x and y coordinates of one view, and either the x or the y coordinates of a second view. Another way to think of this result is that x_1, y_1, x_2 span a 3-D linear subspace in \mathcal{R}^n that includes all sets of x or y coordinates that the object could later produce. We omit a proof of this, but note that the proof is based on a linear transformation.

We now show how a similar result is evident from our work. When one considers the affine coordinates of an image, we have shown that the set of images a model produces occupies a linear subspace, a 2-D plane, of a larger affine space. Our approach also implies a result similar to the one and a half views result described above. Given the α coordinates of any two views of a model, we may determine the line in α space that describes all the α coordinates the model might produce. In fact, any point on this line is a linear combination of the original two points used to determine the line. And since the directions of the α and β lines are the same, if we are given the α coordinates of two images of a model, and the β coordinates of one image, we may also determine the line in β space that describes the model.

We now use our results to show that the linear combinations work cannot be extended to oriented points. To do this it will be sufficient to consider the case where each model consists of three points and three vectors. If the linear combinations result fails in this case, then it fails in general. In this case, we may represent a model's images with a 2-D hyperboloid in a 3-D space. It might seem obvious from this that the linear combinations idea will not apply. Given a 2-D hyperboloid in a 3-D space, it is easy to pick four points on the hyperboloid that span the entire 3-D space. This means that in general, any four images of any model can be linearly combined to produce any possible image, and the linear combinations idea is true only in the trivial sense that with enough images we may express any other image as a linear combination of those images.

However, things are not this simple. Linear combinations might be true of one representation of images, but not true of another. For example, with point features the cartesian coordinates of one image are linear combinations of other images of the same scene, but this might not be true of polar coordinates. So we must prove that the set of all images of a scene are not a linear combination of a small number of images, regardless of our choice of representation for an image. Since we know that the three basis points of the image convey no information about the scene, the real question is whether some alternate representation of affine slope might map each model's images into a linear subspace. So we ask

whether there is a continuous, one-to-one mapping from affine slope space, that is the space defined by $(\theta_0, \theta_1, \theta_2)$, into another space, such that this mapping takes every hyperboloid in affine slope space into a linear subspace. From elementary topology we know that any continuous one-to-one mapping will map our 3-D affine slope space into a space that is also 3-D, and that it will map every 2-D hyperboloid into a 2-D surface. So the question is whether these hyperboloids might map to 2-D planes in a 3-D space?

To answer this, we must look at the particular set of hyperboloids that correspond to possible models. We assume that an appropriate mapping exists for linear combinations, and derive a contradiction. First, we recall that the line $\theta_0 = \theta_1 = \theta_2$ is part of the equation for each hyperboloid corresponding to a possible model. Call this line L . L is a degenerate case; the actual set of images a model produces does not include L , but it includes images that are arbitrarily close to L . Suppose we apply a continuous one-to-one mapping, call it f , that takes one of these hyperboloids, H , to a 2-D plane, $f(H)$. Then $f(L)$ is a 1-D curve such that for any point on the curve, there is a point on $f(H)$ arbitrarily close to that curve point. This can only happen if $f(L)$ lies on $f(H)$. That is, we can omit $f(L)$ from a model's manifold without problems, but if this manifold is linear, then the requirement that our representation be continuous tells us that $f(L)$ must in fact lie in this 2-D plane.

Since L is part of every scene's hyperboloid, this means that $f(L)$ must be a 1-D curve at which all scenes' manifolds intersect, in our new space. If all scenes' manifolds are 2-D planes in this new space, they can only intersect in a line. So $f(L)$ must be a line at which all scenes' planes intersect. But this means that no scenes' planes can intersect anywhere else in our new space. However, we have already shown that in general all the hyperboloids that represent scenes intersect at other places than the line L . f must preserve these intersections, so a contradiction is derived. That is, we have shown that in any space, the manifolds corresponding to two different scenes of oriented points must intersect in two distinct curves, which cannot happen if the manifolds are linear. This tells us that it is never possible to represent the images produced by a scene of oriented points using linear combinations, except in the trivial sense.

The implications of this result, however, depend on what one thinks is important about the linear combinations result. If it is the linearity of the images, then our result concerning oriented points is a setback. It does seem that part of the impact of the linear combinations work is that the linearity of a scene's images was unexpected and striking. And it is certainly true that linear spaces can lead to simpler reasoning than non-linear ones. However, a significant part of the importance of the linear combinations work is that it provides a simple way of characterizing a scene's possible images in terms of a small number of images, without explicitly deriving 3-D information about the scene. And we may still do that with oriented points, as we have shown in our discussion of affine structure from motion. Our compu-

tations are no longer linear, but we may still derive a simple set of equations from a few images of oriented points that characterize all other images that could be produced by the scene, without explicitly determining this scene's 3-D structure.

6 Conclusions

In this paper we have shown how to characterize the sets of images that groups of 3-D oriented point features can produce as simple geometric objects, when a linear transformation is used as a projection method. This means that problems involving matching 2-D images to 3-D scenes can be rephrased as the problem of matching points in a high-dimensional space to 2-D hyperbolic surfaces in that space. This provides us with a formulation of the matching problem that can be easily visualized, and that is easy to reason about. Many problems can be handled very simply, and involve familiar geometric structures residing in a 3-D euclidean space. At the same time, we have demonstrated that when we focus on the topology of this simple space, we can derive results that will apply to any reasonable representation of a model's images.

We use these results to place some fundamental limits on problems in motion understanding and object recognition. We show that although affine structure can be derived simply from a motion sequence, that this derivation requires four images of oriented points. This compares unfavorably with the two images needed to derive the affine structure of simple point features, or the three images needed to derive the rigid structure of simple or oriented point features. Our result therefore undermines one of the motivations for attempting to derive affine structure instead of rigid structure. We also show that indexing oriented points requires us to represent a 2-D surface discretely. Some indexing systems have been built that implicitly represent 2-D surfaces, and that build lookup tables through sampling, but these tend to require considerable space. So it is disappointing to find that for oriented points we cannot find a way to perform indexing by representing 1-D curves, as we did for simple points. Finally, we show that new images of oriented points cannot be constructed from a linear combination of old images, except in a trivial sense. At the same time, we do show how to construct new images from old by solving a simple set of equations. In fact, it is not clear how much of the real value of the linear combinations result is lost, since for oriented points we have a simple method of solving the same basic problem that the linear combinations method solved for simple points. For each of these problems, we have shown that when we add orientation vectors to points, a problem becomes more difficult in an important way, while at the same time we provide positive solutions to these problems.

References

- [1] Basri, R. and Ullman, S., 1988, "The Alignment of Objects with Smooth Surfaces," *Second International Conference on Computer Vision*, pp. 482-488.

- [2] Breuel, T., 1991, "Model Based Recognition using Pruned Correspondence Search," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 257-268.
- [3] Cass, T., 1992, *Polynomial Time Geometric Matching for Object Recognition*, PhD Thesis, MIT Department of Electrical Engineering and Computer Science.
- [4] Clemens, D. and Jacobs, D., 1991, "Space and Time Bounds on Model Indexing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):1007-1018.
- [5] Jacobs, D., 1992, "Space Efficient 3D Model Indexing," *IEEE Conference Computer on Vision and Pattern Recognition*, pp. 439-444.
- [6] Jacobs, D., 1992, "Recognizing 3-D Objects Using 2-D Images," PhD. Thesis, MIT Department of Electrical Engineering and Computer Science.
- [7] Koenderink, J. and van Doorn, A., 1991, "Affine Structure from Motion," *J. of the Optical Soc. of America*, 8(2):377-385.
- [8] Korn, G.A. & T.M. Korn, 1968, *Mathematical Handbook for Scientists and Engineers*, McGraw-Hill, New York.
- [9] Lamdan, Y. & H.J. Wolfson, 1988, "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme," *Second Int. Conference Computer Vision*, pp. 238-249.
- [10] Poggio, T., 1990, "3D Object Recognition: On a Result of Basri and Ullman," Istituto Per La Ricerca Scientifica E Tecnologica IRST Technical Report 9005-03.
- [11] Roberts, L., 1966, "Machine Perception of Three-Dimensional Solid Objects," *Optical and Electr-optical Information Processing*, edited by J. Tippett, MIT Press, Cambridge.
- [12] Shashua, A., 1991, "Correspondence and Affine Shape from Two Orthographic Views: Motion and Recognition," MIT AI Memo 1327.
- [13] Thompson, D. & J.L. Mundy, 1987, "Three-Dimensional Model Matching From an Unconstrained Viewpoint", *Proc. IEEE Conference Rob. Aut.*, pp. 208-220.
- [14] Ullman, S., 1979, *The Interpretation of Visual Motion*, MIT Press, Cambridge.
- [15] Ullman, S. and Basri, R., 1991, "Recognition by Linear Combinations of Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):992-1007.
- [16] Van Gool, L., P. Kempenaers & A. Oosterlinck, 1991, "Recognition and Semi-Differential Invariants," *IEEE Conference Computer Vision and Pattern Recognition*, pp. 454-460.

Localization and Positioning using Combinations of Model Views

Ronen Basri

Dept. of Applied Math.
The Weizmann Inst. of Science
Rehovot, 76100
Israel

Ehud Rivlin

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742

Abstract

A method for localization and positioning in an indoor environment is presented. *Localization* is the act of recognizing the environment, and *positioning* is the act of computing the exact coordinates of a robot in the environment. The method is based on representing the scene as a set of 2D views and predicting the appearances of novel views by linear combinations of the model views. The method accurately approximates the appearance of scenes under weak perspective projection. Analysis of this projection as well as experimental results demonstrate that in many cases this approximation is sufficient to accurately describe the scene. When weak perspective approximation is invalid, either a larger number of models can be acquired or an iterative solution to account for the perspective distortions can be employed. The same principal method is applied for both the localization and positioning problems, and a simple algorithm for *repositioning*, the task of returning to a previously visited position defined by a single view, is derived from this method.

1 Introduction

Basic tasks in autonomous robot navigation are localization and positioning. *Localization* is the act of recognizing the environment, that is, assigning consistent labels to different locations, and *positioning* is the act of computing the coordinates of the robot in the environment. Positioning is a task complementary to localization, in the sense that position (e.g., "1.5 meters northwest of table T ") is often specified in a place-specific coordinate system ("in room 911"). In this paper we suggest a method of both localization and positioning using vision alone. A variant of the positioning problem, referred

to as *repositioning*, involving the return to a previously visited place is also discussed.

Previous studies have examined the problems of localization and positioning under a variety of conditions, defined by the kind of sensors employed, the nature of the environment, and the representations used. We can distinguish between active and passive sensing, indoor and outdoor navigation tasks, and metric and topological representations. The metric approach attempts to utilize a detailed geometric description of the environment, while the topological approach uses a more qualitative description including a graph with nodes representing places and arcs representing sequences of actions that would result in moving the robot from one node to another.

In the paper we consider a robot that uses a passive sensor, vision, in an indoor environment. The paper addresses both the localization and the positioning problems. Solutions to these problems are presented based on object recognition techniques. The method, based on the linear combinations scheme [18], represents scenes by sets of their 2D images. Localization is achieved by comparing the observed image to linear combinations of model views, and the position of the robot is computed by analyzing the coefficients of the linear combination that aligns the model to the image.

The rest of the paper is organized as follows. The next section describes the localization and positioning problems and surveys previous solutions. The method of localization and positioning using linear combinations of model views is described in Section 3. The method assumes weak perspective projection. An iterative scheme to account for perspective distortions is presented in Section 4. An analysis of the error resulting from the projection assumption is presented in Section 5. Constraints imposed on the motion of the robot as a result of special properties of indoor environments can be used to reduce the complexity of the method presented here. Experimental results follow.

2 The Problem

Localization and positioning from visual input are defined in the following way: Given a familiar environment, identify the observed environment, and then find your position in that environment. Localization resembles the task of object recognition, with objects replaced

⁰R. B. was supported in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-91-J-4038. E. R. was supported in part by the Defense Advanced Research Projects Agency (ARPA Order No. 8459) and the U.S. Army Engineer Topographic Laboratories under Contract DACA76-92-C-0009.

by scenes. Once localization is accomplished, positioning can be performed.

One problem a system for localization and positioning should address is the variability of images due to viewpoint changes. The inexactness of practical systems makes it difficult for a robot to return to a specified position on subsequent visits. The visual data available to the robot between visits varies in accordance with the viewing position of the robot. A localization system should be able to recognize scenes from different positions and orientations.

Another problem is that of changes in the scene. At subsequent visits the same place may look different due to changes in the arrangement of the objects, the introduction of new objects, and the removal of others. In general, some objects tend to be more static than others. While chairs and books are often moved, tables, closets, and pictures tend to change their position much less, and walls are almost guaranteed to be static. Static cues naturally are more reliable than mobile ones. Confining the system to static cues, however, may in some cases result in failure to recognize the scene due to insufficient cues. The system should therefore attempt to rely on static cues, but should not ignore the dynamic cues.

Solutions to the problem of localization from visual data require a large memory and heavy computation. Existing systems often try to reduce this cost by using sparse representations and by exploiting contextual information. Sparse representations are introduced in [10, 15]. Mataric [10] represents scenes as sequences of landmarks (such as walls, doors, etc.) extracted by tracing the boundaries of the scene using a sonar and a compass. Metric information of and between the landmarks is not stored. Sarachik [15] recognizes a room by its dimensions, which are measured by identifying and locating the top corners of the room using stereo data (obtained from four cameras). In both cases the representation is very sparse, and the scene is therefore often ambiguous.

Richer representations are used in [3, 5] where higher success rates are reported. Braunegg [3] represents the scene by an occupancy table, a 2D bit array which contains a 1 at every location occupied by some object. The table is constructed by taking stereo pictures covering 360° from the middle of the room and projecting the obtained 3D data onto the floor. The method suffers from loss of information due to the projection onto the floor.

Engelson *et al.* [5] represent the scene by a set of invariant "signatures". A signature is usually composed of low-resolution gray-level or range data obtained by blurring an image. A set of signatures taken from different viewpoints are stored. A scene is recognized if the robot encounters a signature similar to one of the stored signatures.

Systems that use the full information provided by the image (e.g., [6, 12]) usually rely on contextual information to avoid scanning all the models in the memory and to reduce the computational cost of comparing a model to the image. The system follows a predetermined path, so that the identity of each visited location is known in

advance, and localization becomes a verification problem. Path continuity in many cases is essential, and the so-called "drop-off" problem is not addressed. The emphasis in these systems is on positioning, which is used to keep the robot on the path. It is typical for these systems (e.g., [1, 6, 12]) to use a full 3D model of the environment.

Onoguchi *et al.* [12], among others, represent the environment by a set of landmarks selected from pairs of stereo images by a human operator. These landmarks are transformed by an image processing program which is designed so as to identify the specific landmark using specific extraction instructions (such as what features to look for and at what locations). Localization is achieved by applying the extraction procedure specified for the next landmark. Once a landmark is identified, the position of the robot relative to that landmark is determined by comparing the dimensions of the observed landmark with those of the stored model.

The method presented in this paper represents the environment using a set of views given as edge maps. Localization and positioning are achieved by comparing images of the environment to linear combinations of the model views. The method uses rich visual information to represent the scene. The system is flexible. In many cases it is capable of recognizing its location from one image only (360° coverage is not required). When one image is not sufficient, additional images can be acquired to solve the localization problem. Context can be used to determine the order of comparison of the models to the observed image and to increase the confidence of a given match, but context is not essential: the system can also, by performing more extensive computations, solve the "drop-off" problem.

3 The Method

The problems of localization and object recognition are similar in many ways. Both problems require the matching of visual images to stored models, either of the environment or of the observed objects. Both problems face similar difficulties, such as varying illumination conditions and changes in appearance due to viewpoint changes. Similar methodologies therefore can be used for both problems.

A particular application of an object recognition scheme, the Linear Combinations (LC) scheme [18], to the problems of localization and positioning is discussed below. The environment is represented in this scheme by a small set of views obtained from different viewpoints and by the correspondence between the views. A novel view is recognized by comparing it to linear combinations of the stored views. Positioning is achieved by recovering the position of the camera relative to its position in the model views from the coefficients of the aligning linear combination. In the rest of this section we review the linear combinations approach and describe its application to both localization and positioning.

3.1 Localization

The problem of localization is defined as follows: given P , a 2D image of a place, and \mathcal{M} , a set of stored mod-

els, find a model $M^i \in \mathcal{M}$ such that P matches M^i . A common approach to handling the problem of recognition from different viewpoints is by comparing the stored models to the observed environment after the viewpoint is recovered and compensated for. This approach, called *alignment*, is used in a number of studies of object recognition [2, 7, 8, 9, 16, 17]. We apply the alignment approach to the problem of localization. The system described below uses the "Linear Combinations" (LC) scheme, which was suggested by Ullman and Basri [18].

We begin with a brief review of the LC scheme. LC is defined as follows. Given an image, we construct two view vectors from the feature points in the image, one contains the x -coordinates of the points, and the other contains the y -coordinates of the points. An object (in our case, the environment) is modeled by a set of such views, where the points in these views are ordered in correspondence. The appearance of a novel view of the object is predicted by applying linear combinations to the stored views. The predicted appearance is then compared with the actual image, and the object is recognized if the two match. The advantage of this method is twofold. First, viewer-centered representations are used rather than object-centered ones, namely, models are composed of 2D views of the observed scene; second, novel appearances are predicted in a simple and accurate way (under weak perspective projection).

Formally, given P , a 2D image of a scene, and \mathcal{M} , a set of stored models, the objective is to find a model $M^i \in \mathcal{M}$ such that $P = \sum_{j=1}^k \alpha_j M_j^i$ for some constants $\alpha_j \in \mathcal{R}$. It has been shown that this scheme accurately predicts the appearance of rigid objects under weak perspective projection (orthographic projection and scale). The limitations of this projection model are discussed later in this paper.

More concretely, let $p_i = (x_i, y_i, z_i)$, $1 \leq i \leq n$, be a set of n object points. Under weak perspective projection, the position $p'_i = (x'_i, y'_i)$ of these points in the image are given by

$$\begin{aligned} x'_i &= sr_{11}x_i + sr_{12}y_i + sr_{13}z_i + t_x \\ y'_i &= sr_{21}x_i + sr_{22}y_i + sr_{23}z_i + t_y \end{aligned} \quad (1)$$

where r_{ij} are the components of a 3×3 rotation matrix, and s is a scale factor. Rewriting this in vector equation form we obtain

$$\begin{aligned} \mathbf{x}' &= sr_{11}\mathbf{x} + sr_{12}\mathbf{y} + sr_{13}\mathbf{z} + t_x\mathbf{1} \\ \mathbf{y}' &= sr_{21}\mathbf{x} + sr_{22}\mathbf{y} + sr_{23}\mathbf{z} + t_y\mathbf{1} \end{aligned} \quad (2)$$

where $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{x}', \mathbf{y}' \in \mathcal{R}^n$ are the vectors of x_i, y_i, z_i, x'_i and y'_i coordinates respectively, and $\mathbf{1} = (1, 1, \dots, 1)$. Consequently,

$$\mathbf{x}', \mathbf{y}' \in \text{span}\{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{1}\} \quad (3)$$

or, in other words, \mathbf{x}' and \mathbf{y}' belong to a four-dimensional linear subspace of \mathcal{R}^n . (Notice that \mathbf{z}' , the vector of depth coordinates of the projected points, also belongs to this subspace. This fact is used in Section 4 below.) A four-dimensional space is spanned by any four linearly independent vectors of the space. Two views of the scene supply four such vectors [13, 18]. Denote by

$\mathbf{x}_1, \mathbf{y}_1$ and $\mathbf{x}_2, \mathbf{y}_2$ the location vectors of the n points in the two images; then there exist coefficients a_1, a_2, a_3, a_4 and b_1, b_2, b_3, b_4 such that

$$\begin{aligned} \mathbf{x}' &= a_1\mathbf{x}_1 + a_2\mathbf{y}_1 + a_3\mathbf{x}_2 + a_4\mathbf{1} \\ \mathbf{y}' &= b_1\mathbf{x}_1 + b_2\mathbf{y}_1 + b_3\mathbf{x}_2 + b_4\mathbf{1} \end{aligned} \quad (4)$$

(Note that the vector \mathbf{y}_2 already depends on the other four vectors.) Since R is a rotation matrix, the coefficients satisfy the following two quadratic constraints:

$$\begin{aligned} a_1^2 + a_2^2 + a_3^2 - b_1^2 - b_2^2 - b_3^2 &= \\ 2(b_1b_3 - a_1a_3)r_{11} + 2(b_2b_3 - a_2a_3)r_{12} & \end{aligned} \quad (5)$$

and

$$\begin{aligned} a_1b_1 + a_2b_2 + a_3b_3 + (a_1b_3 + a_3b_1)r_{11} + \\ (a_2b_3 + a_3b_2)r_{12} &= 0 \end{aligned} \quad (6)$$

To derive these constraints the transformation between the two model views should be recovered. This can be done under weak perspective using a third image. Alternatively, the constraints can be ignored, in which case the system would confuse rigid transformations with affine ones. This usually does not prevent successful localization since generally scenes are fairly different from one another.

The LC scheme for the problem of localization is as follows. The environment is modeled by a set of images with correspondence between the images. For example, a spot can be modeled by two of its corresponding views. The corresponding quadratic constraints may also be stored. Localization is achieved by recovering the linear combination that aligns the model to the observed image. The coefficients are determined using four model points and their corresponding image points by solving a linear set of equations. Three points are sufficient to determine the coefficients if the quadratic constraints are also considered. Additional points may be used to reduce the effect of noise.

The LC scheme uses viewer-centered models, that is, representations that are composed of images. It has a number of advantages over methods that build full three-dimensional models to represent the scene. First, by using viewer-centered models that cover relatively small transformations we avoid the need to handle occlusions in the scene. If from some viewpoints the scene appears different because of occlusions we utilize a new model for these viewpoints. Second, viewer-centered models are easier to build and to maintain than object-centered ones. The models contain only images and correspondences. By limiting the transformation between the model images one can find the correspondence using motion methods. If large portions of the environment are changed between visits a new model can be constructed by simply replacing old images with new ones.

One problem with using the LC scheme for localization is due to the weak perspective approximation. In contrast with the problem of object recognition, where we can generally assume that objects are small relative to their distance from the camera, in localization the environment surrounds the robot and perspective distortions cannot be neglected. The limitations of weak perspective

modeling are discussed both mathematically and empirically in the next two sections. It is shown that in many practical cases weak perspective is sufficient to enable accurate localization. The main reason is that the problem of localization does not require accurate measurements in the entire image; it only requires identifying a sufficient number of spots to guarantee accurate naming. If these spots are relatively close to the center of the image, or if the depth differences they create are relatively small (as in the case of looking at a wall when the line of sight is nearly perpendicular to the wall), the perspective distortions are relatively small, and the system can identify the scene with high accuracy. Also, views related by a translation parallel to the image plane form a linear space even when perspective distortions are large.

By using weak perspective we avoid stability problems that frequently occur in perspective computations. We can therefore compute the alignment coefficients by looking at a relatively narrow field of view. The entire scheme can be viewed as an accumulative process. Rather than acquiring images of the entire scene and comparing them all to a full scene model (as in [3]) we recognize the scene image by image, spot by spot, until we accumulate sufficient convincing information that indicates the identity of the place.

When perspective distortions are relatively large and weak perspective is insufficient to model the environment, two approaches can be used. One possibility is to construct a larger number of models so as to keep the possible changes between the familiar and the novel views small. Alternatively, an iterative computation can be applied to compensate for these distortions. Such an iterative method is described in Section 4.

3.2 Positioning

Positioning is the problem of recovering the exact position of the robot. This position can be specified in a fixed coordinate system associated with the environment (i.e., room coordinates), or it can be associated with some model, in which case location is expressed with respect to the position from which the model views were acquired. In this section we discuss an application of the LC scheme to the positioning problem.

The idea is the following. We assume a model composed of two images, P_1 and P_2 ; their relative position is given. Given a novel image P' , we first align the model with the image (i.e., localization). By considering the coefficients of the linear combination the robot's position relative to the model images is recovered. To recover the absolute position of the robot in the room the absolute positions of the model views should also be provided.

Assuming P_2 is obtained from P_1 by a rotation R , translation $t = (t_x, t_y)$, and scaling s , the coordinates of a point in P' , (x', y') , can be written as linear combinations of the corresponding model points in the following way:

$$\begin{aligned} x' &= a_1 x_1 + a_2 y_1 + a_3 x_2 + a_4 \\ y' &= b_1 x_1 + b_2 y_1 + b_3 x_2 + b_4 \end{aligned} \quad (7)$$

Substituting for x_2 we can derive the position of the robot in the image relative to its position in the model

views:

$$\begin{aligned} \Delta x &= a_3 t_x + a_4 \\ \Delta y &= b_3 t_y + b_4 \\ \Delta z &= f\left(\frac{1}{s_n} - \frac{1}{s}\right) \end{aligned} \quad (8)$$

where

$$s_n^2 = a_1^2 + a_2^2 + a_3^2 s^2 + 2a_3 s(a_1 r_{11} + a_2 r_{12}) \quad (9)$$

Note that Δz is derived from the change in scale of the object. The rotation of the camera can also be derived (see details in [14]).

As was already mentioned, the position of the robot is computed here relative to the position of the camera when the first model image, P_1 , was acquired. Δx and Δz represent the motion of the robot from P_1 to P' , and the rest of the parameters represent its 3D rotation and elevation. To obtain the relative position the transformation parameters between the model views, P_1 and P_2 , are required.

3.3 Repositioning

An interesting variant of the positioning problem, referred to as *repositioning*, is defined as follows. Given an image, called the *target* image, position yourself in the location from which this image was observed¹. One way to solve this problem is to extract the exact position from which the target image was obtained and direct the robot to that position. In this section we are interested in a more qualitative approach. Under this approach position is not computed. Instead, the robot observes the environment and extracts only the direction to the target location. Unlike the exact approach, the method presented here does not require the recovery of the transformation between the model views.

We assume we are given with a model of the environment together with a target image. The robot is allowed to take new images as it is moving towards the target. We assume a horizontally moving platform. (In other words, we assume three degrees of freedom rather than six; the robot is allowed to rotate around the vertical axis and translate horizontally.) Below we give a simple computation that determines a path which terminates in the target location. At each time step the robot acquires a new image and aligns it with the model. By comparing the alignment coefficients with the coefficients for the target image the robot determines its next step. The algorithm is divided into two stages. In the first stage the robot fixates on one identifiable point and moves along a circular path around the fixation point until the line of sight to this point coincides with the line of sight to the corresponding point in the target image. In the second stage the robot advances forward or retreats backward until it reaches the target location.

Given a model composed of two images, P_1 and P_2 , P_2 is obtained from P_1 by a rotation about the Y -axis by an angle α , horizontal translation t_x , and scale factor

¹This problem can be considered as a variant of the homing problem. A discussion of the general homing problem with a "signature-based" solution can be found in [11].

s. Given a target image P_t , P_t is obtained from P_1 by a similar rotation by an angle θ , translation t_t , and scale s_t . Using Eq. (4) the position of a target point (x_t, y_t) can be expressed as

$$\begin{aligned} x_t &= a_1 x_1 + a_3 x_2 + a_4 \\ y_t &= b_2 y_1 \end{aligned} \quad (10)$$

(The rest of the coefficients are zero since the platform moves horizontally.) In fact, the coefficients are given by

$$\begin{aligned} a_1 &= \frac{s_t \sin(\alpha - \theta)}{\sin \alpha} & a_3 &= \frac{s_t \sin \theta}{s \sin \alpha} \\ a_4 &= t_t - \frac{t_x s_t \sin \theta}{s \sin \alpha} & b_2 &= s_t \end{aligned} \quad (11)$$

(The derivation is given in [14])

At every time step the robot acquires an image and aligns it with the above model. Assume that image P_p is obtained as a result of a rotation by an angle ϕ , translation t_p , and scale s_p . The position of a point (x_p, y_p) is expressed by

$$\begin{aligned} x_p &= c_1 x_1 + c_3 x_2 + c_4 \\ y_p &= d_2 y_1 \end{aligned} \quad (12)$$

where the coefficients are given by

$$\begin{aligned} c_1 &= \frac{s_p \sin(\alpha - \phi)}{\sin \alpha} & c_3 &= \frac{s_p \sin \phi}{s \sin \alpha} \\ c_4 &= t_p - \frac{t_x s_p \sin \phi}{s \sin \alpha} & d_2 &= s_p \end{aligned} \quad (13)$$

The step performed by the robot is determined by

$$\delta = \frac{c_1}{c_3} - \frac{a_1}{a_3} \quad (14)$$

That is,

$$\delta = s \sin \alpha (\cot \phi - \cot \theta) \quad (15)$$

δ is a monotonic function of the angular difference between P_p and P_t (the derivation is given in [14]). The robot should now move so as to reduce the absolute value of δ . The direction of motion depends on the sign of δ . The robot can deduce the direction by moving slightly to the side and checking if this motion results in an increase or decrease of δ . The motion is defined as follows. The robot moves to the right (or to the left, depending on which direction reduces $|\delta|$) by a step Δx .

A new image P_n is now acquired, and the fixated point is located in this image. Denote its new position by x_n . Since the motion is parallel to the image plane the depth values of the point in the two views, P_p and P_n , are identical. We now want to rotate the camera so as to return the fixated point to its original position. The angle of rotation, β , can be deduced from the equation

$$x_p = x_n \cos \beta + \sin \beta \quad (16)$$

This equation has two solutions. We chose the one that counters the translation (namely, if translation is to the right, the camera should rotate to the left), and that keeps the angle of rotation small. In the next time step the new picture P_n replaces P_p and the procedure is repeated until δ vanishes. The resulting path is circular around the point of focus.

Once the robot arrives at a position for which $\delta = 0$ (namely, its line of sight coincides with that of the target image, and $\phi = \theta$) it should now advance forward or

retreat backward to adjust its position along the line of sight. Several measures can be used to determine the direction of motion; one example is the term c_1/a_1 which satisfies

$$\frac{c_1}{a_1} = \frac{s_p}{s_t} \quad (17)$$

when the two lines of sight coincide. The objective at this stage is to bring this measure to 1.

4 Handling Perspective Distortions

The linear combination scheme presented above accurately handles changes in viewpoint assuming the images are obtained under weak perspective projection. Error analysis and experimental results demonstrate that in many practical cases this assumption is valid. In cases where perspective distortions are too large to be handled by a weak perspective approximation, matching between the model and the image can be facilitated in two ways. One possibility is to avoid cases of large perspective distortion by augmenting the library of stored models with additional models. In a relatively dense library there usually exists a model that is related to the image by a sufficiently small transformation avoiding such distortions. The second alternative is to improve the match between the model and the image using an iterative process. In this section we consider the second option.

The suggested iterative process is based on a Taylor expansion of the perspective coordinates. As described below, this expansion results in a polynomial consisting of terms each of which can be approximated by linear combinations of views. The first term of this series represents the orthographic approximation. The process resembles a method of matching 3D points with 2D points described recently by DeMenthon and Davis [4]. In this case, however, the method is applied to 2D models rather than 3D ones. In our application the 3D coordinates of the model points are not provided; instead they are approximated from the model views.

An image point $(x, y) = (fX/Z, fY/Z)$ is the projection of some object point, (X, Y, Z) in the image, where f denotes the focal length. Consider the following Taylor expansion of $F(Z) = 1/Z$ around some depth value Z_0 :

$$\frac{1}{Z} = \sum_{k=0}^{\infty} \frac{F^{(k)}(Z_0)}{k!} (Z - Z_0)^k \quad (18)$$

The Taylor series describing the position of a point $x = fX/Z$ therefore is given by

$$x = \frac{fX}{Z_0} \left[1 + \sum_{k=1}^{\infty} \frac{(-1)^k}{(k-1)!} \left(\frac{Z - Z_0}{Z_0} \right)^k \right] \quad (19)$$

Notice that the zero term contains the orthographic approximation for x . Denote by $\Delta^{(k)}$ the k th term of the series:

$$\Delta^{(k)} = \frac{fX}{Z_0} \frac{(-1)^k}{(k-1)!} \left(\frac{Z - Z_0}{Z_0} \right)^k \quad (20)$$

A recursive definition of the above series is given below.

Initialization:

$$\mathbf{x}^{(0)} = \Delta^{(0)} = \frac{fX}{Z_0}$$

Iterative step:

$$\begin{aligned}\Delta^{(k)} &= -\frac{Z - Z_0}{(k-1)Z_0} \Delta^{(k-1)} \\ \mathbf{x}^{(k)} &= \mathbf{x}^{(k-1)} + \Delta^{(k)}\end{aligned}$$

where $\mathbf{x}^{(k)}$ represents the k th order approximation for \mathbf{x} , and $\Delta^{(k)}$ represents the highest order term in $\mathbf{x}^{(k)}$.

According to the orthographic approximation both X and Z can be expressed as linear combinations of the model views (Eq. (4)). We therefore apply the above procedure, approximating X and Z at every step using the linear combination that best aligns the model points with the image points. The general idea is therefore the following. First, we estimate $\mathbf{x}^{(0)}$ and $\Delta^{(0)}$ by solving the orthographic case. Then at each step of the iteration we improve the estimate by seeking the linear combination that best estimates the factor

$$-\frac{Z - Z_0}{(k-1)Z_0} \approx \frac{\mathbf{x} - \mathbf{x}^{(k-1)}}{\Delta^{(k-1)}} \quad (21)$$

Denote by $\mathbf{x} \in \mathcal{R}^n$ the vector of image point coordinates, and denote by

$$P = [\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, 1] \quad (22)$$

an $n \times 4$ matrix containing the position of the points in the two model images. Denote by $P^+ = (P^T P)^{-1} P^T$ the pseudo-inverse of P (we assume P is overdetermined). Also denote by $\mathbf{a}^{(k)}$ the coefficients computed for the k th step. $P\mathbf{a}^{(k)}$ represents the linear combination computed at that step to approximate the X or the Z values. Since at every step Z_0 , f , and k are constant they can be merged into the linear combination. Denote by $\mathbf{x}^{(k)}$ and $\Delta^{(k)}$ the vectors of computed values of \mathbf{x} and Δ at the k th step. An iterative procedure to align a model to the image is described below.

Initialization:

Solve the orthographic approximation, namely

$$\begin{aligned}\mathbf{a}^{(0)} &= P^+ \mathbf{x} \\ \mathbf{x}^{(0)} = \Delta^{(0)} &= P\mathbf{a}^{(0)}\end{aligned}$$

Iterative step:

$$\begin{aligned}\mathbf{q}^{(k)} &= (\mathbf{x} - \mathbf{x}^{(k-1)}) \div \Delta^{(k-1)} \\ \mathbf{a}^{(k)} &= P^+ \mathbf{q}^{(k)} \\ \Delta^{(k)} &= (P\mathbf{a}^{(k)}) \otimes \Delta^{(k-1)} \\ \mathbf{x}^{(k)} &= \mathbf{x}^{(k-1)} + \Delta^{(k)}\end{aligned}$$

where the vector operations \otimes and \div are defined as

$$\begin{aligned}\mathbf{u} \otimes \mathbf{v} &= (u_1 v_1, \dots, u_n v_n) \\ \mathbf{u} \div \mathbf{v} &= \left(\frac{u_1}{v_1}, \dots, \frac{u_n}{v_n} \right)\end{aligned}$$

$x \setminus \epsilon$	5	10	15	20
25	1.2	1.4	1.6	1.8
50	1.1	1.2	1.3	1.4
75	1.07	1.13	1.2	1.27
100	1.05	1.1	1.15	1.2

Table 1: Allowed depth ratios, $\frac{Z}{Z_0}$, as a function of x (half the width of the field considered) and the error allowed (ϵ , in pixels).

5 Projection Model—Error Analysis

In this section we estimate the error obtained by using the linear combination method. The method assumes a weak perspective projection model. We compare this assumption with the more accurate perspective projection model.

A point (X, Y, Z) is projected under the perspective model to the point $(x, y) = (fX/Z, fY/Z)$ in the image, where f denotes the focal length. Under the weak perspective model the same point is approximated by $(\hat{x}, \hat{y}) = (sX, sY)$ where s is a scaling factor. The best estimate for s , the scaling factor, is given by $s = f/Z_0$, where Z_0 is the average depth of the observed environment. Denote the error by

$$E = |\hat{x} - x| \quad (23)$$

The error is expressed by

$$E = \left| fX \left(\frac{1}{Z_0} - \frac{1}{Z} \right) \right| \quad (24)$$

Changing to image coordinates we obtain

$$E = |x| \left| \frac{Z}{Z_0} - 1 \right| \quad (25)$$

The error is small when the measured feature is close the optical axis, or when our estimate for the depth, Z_0 , is close to the real depth, Z . This supports the basic intuition that for images with low depth variance and for fixated regions (regions near the center of the image), the obtained perspective distortions are relatively small, and the system can therefore identify the scene with high accuracy. Table 1 shows a number of examples where the allowed depth variance, Z/Z_0 , is computed as a function of x and the tolerated error, ϵ . For example, a 10 pixel error tolerated in a field of view of up to ± 50 pixels is equivalent to allowing depth variations of 20%. From this discussion it is apparent that when a model is aligned to the image the results of this alignment should be judged differently at different points of the image. The farther away a point is from the center the more discrepancy should be tolerated between the prediction and the actual image. A five pixel error at position $x = 50$ is equivalent to a 10 pixel error at position $x = 100$.

So far we have considered the discrepancies between the weak perspective and the perspective projections of points. The accuracy of the LC scheme depends on the validity of the weak perspective projection both in the

model views and for the incoming image. In the rest of this section we develop an error term for the LC scheme assuming that both the model views and the incoming image are obtained by perspective projection.

The error obtained by using the LC scheme is given by

$$E = |x - ax_1 - by_1 - cx_2 - d| \quad (26)$$

Since the scheme accurately predicts the appearances of points under weak perspective projection, it satisfies

$$\hat{x} = a\hat{x}_1 - b\hat{y}_1 - c\hat{x}_2 - d \quad (27)$$

where accented letters represent orthographic approximations. Assume that in the two model pictures the depth ratios are roughly equal:

$$\frac{Z_0^M}{Z^M} = \frac{Z_{01}}{Z_1} \approx \frac{Z_{02}}{Z_2} \quad (28)$$

(This condition is satisfied, for example, when between the two model images the camera only translates along the image plane.) Using the fact that

$$x = \frac{fX}{Z} = \frac{fX}{Z_0} \frac{Z_0}{Z} = \hat{x} \frac{Z_0}{Z} \quad (29)$$

we obtain (see [14])

$$E \leq |\hat{x}| \left| \frac{Z_0}{Z} - \frac{Z_0^M}{Z^M} \right| + |d| \left| \frac{Z_0^M}{Z^M} - 1 \right| \quad (30)$$

The error therefore depends on two terms. The first gets smaller as the image points get closer to the center of the frame and as the difference between the depth ratios of the model and the image gets smaller. The second gets smaller as the translation component gets smaller and as the model gets close to orthographic.

Following this analysis, weak perspective can be used as a projection model when the depth variations in the scene are relatively low and when the system concentrates on the center part of the image. We conclude that, by fixating on distinguished parts of the environment, the linear combinations scheme can be used for localization and positioning.

6 Experiments

The LC method was implemented and applied to images taken in an indoor environment. Images of two offices, A and B, that have similar structures were taken using a Panasonic camera with a focal length of 700 pixels. Semi-static objects, such as heavy furniture and pictures, were used to distinguish between the offices. Figure 1 shows two model views of office A. The views were taken at a distance of about 4m from the wall. Correspondences were picked manually. The results of aligning the model views to images of the two offices are presented in Figure 2. The left image contains an overlay of a predicted image (the thick white lines), constructed by linearly combining the two views, and an actual image of office A. A good match between the two was achieved. The right image contains an overlay of a predicted image constructed from a model of office B and an image of office A. Because the offices share a similar structure the

static cues (the wall corners) were perfectly aligned. The semi-static cues, however, did not match any features in the image.

Figure 3 shows the matching of the model of office A with an image of the same office obtained by a relatively large motion forward (about 2m) and to the side (about 1.5m). Although the distances are relatively short most perspective distortions are negligible, and a good match between the model and the image is obtained.

Another set of images was taken in a corridor. Here, because of the deep structure of the corridor, perspective distortions are noticeable. Nevertheless, the alignment results still demonstrate an accurate match in large portions of the image. Figure 4 shows to model images of a corridor. Figure 5 (left) shows an overlay of a linear combination of the model views with an image of a corridor. It can be seen that the parts that are relatively distant align perfectly. Figure 5 (right) shows the matching of the corridor model with an image obtained by a relatively large motion (about half of the corridor length). Because of perspective distortions the relatively near features no longer align (e.g., the near door edges). The relatively far edges, however, still match.

The next experiment shows the application of the iterative process presented in Section 4 in cases where large perspective distortion were noticeable. Figure 6 shows the results of matching an office model to an image of the same office. In this case, since the image was taken from a relatively close distance, perspective distortions cannot be neglected. The effects of perspective distortions can be noticed on the top right corner of the board, and on the edges of the hanger on the top right. Perspective effects were reduced by using the iterative process. The results of applying this procedure after one and three iterations are shown in Figure 7.

The experimental results demonstrate that the LC method achieves accurate localization in many cases, and that when the method fails because of large perspective distortions an iterative computation can be used to improve the quality of the match.

7 Conclusions

A method of localization and positioning in an indoor environment was presented. The method is based on representing the scene as a set of 2D views and predicting the appearance of novel views by linear combinations of the model views. The method accurately approximates the appearances of scenes under weak perspective projection. Analysis of this projection as well as experimental results demonstrate that in many cases this approximation is sufficient to accurately describe the scene. When the weak perspective approximation is invalid, either a larger number of models can be acquired or an iterative solution can be employed to account for the perspective distortions.

The method presented in this paper has several advantages over existing methods. It uses relatively rich representations; the representations are 2D rather than 3D, and localization can be done from a single 2D view only. The same basic method is used in both the localization and positioning problems, and a simple algorithm



Figure 1: Two model views of office A.

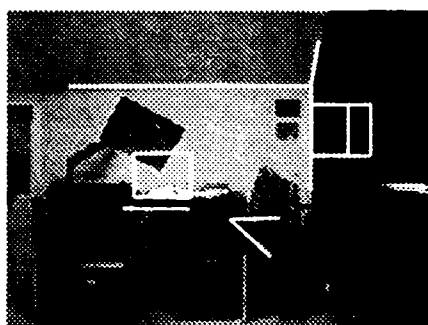
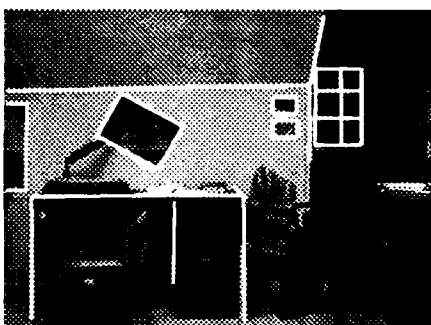


Figure 2: Matching a model of office A to an image of office A (left), and matching a model of office B to the same image (right)

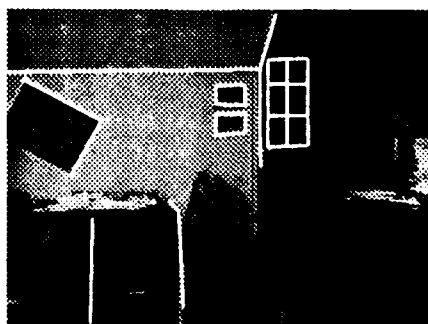


Figure 3: Matching a model of office A to an image of the same office obtained by a relatively large motion forward and to the right.

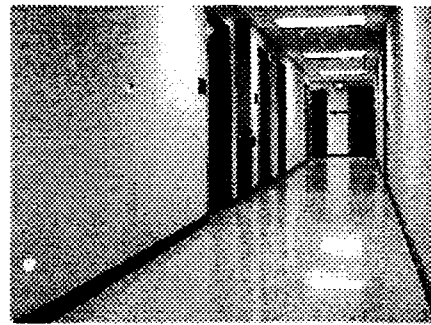


Figure 4: Two model views of a corridor.

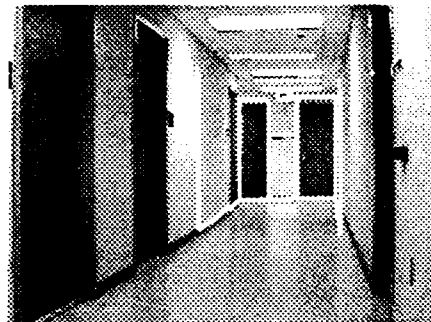
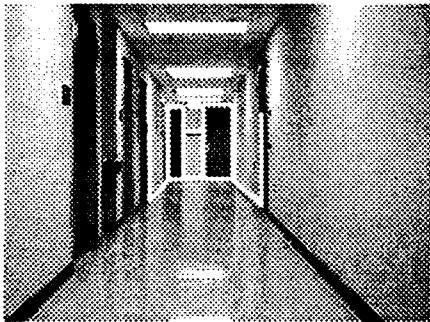


Figure 5: Matching the corridor model with two images of the corridor. The right image was obtained by a relatively large motion forward (about half of the corridor length) and to the right.

for repositioning is derived from this method. Future work includes handling the problem of acquisition and maintenance of models, developing efficient and robust algorithms for solving the correspondence problem, and building maps using visual input.

References

- [1] N. Ayache and O. D. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Trans. on Rob. and Auto.*, 5, 804-819, 1989.
- [2] R. Basri and S. Ullman. The alignment of objects with smooth surfaces. *Proc. 2nd Int. Conf. on Computer Vision*, Tarpon Springs, FL, 482-488, 1988.
- [3] D. J. Braunegg. Marvel-A system for recognizing world locations with stereo vision. *AI-TR-1229*, MIT, 1990.
- [4] D. F. DeMenthon and L. S. Davis. Model-based object pose in 25 lines of code. *Proc. 2nd European Conf. on Comp. Vis.*, Genova, Italy, 1992.
- [5] S. P. Engelson and D. V. McDermott. Image signatures for place recognition and map construction. *Proc. SPIE Symposium on Intel. Rob. Sys.*, Boston, MA, 1991.
- [6] C. Fennema, A. Hanson, E. Riseman, R. J. Beveridge, and R. Kumar. Model-directed mobile robot navigation. *IEEE Trans. on Sys., Man and Cybern.*, 20, 1352-1369, 1990.
- [7] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Com. of the ACM*, 24, 381-395, 1981.
- [8] D. P. Huttenlocher and S. Ullman. Object recognition using alignment. *Proc. 1st Int. Conf. on Comp. Vis.*, London, UK, 102-111, 1987.
- [9] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Rob. Res. TR 202*, Courant Inst. of Math. Sci., NYU, 1985.

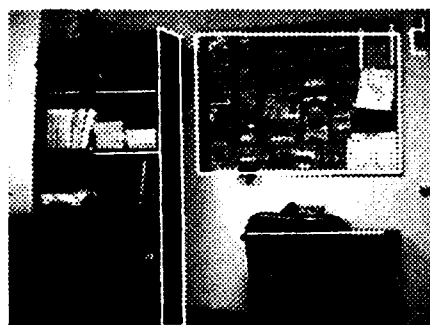


Figure 6: The result of matching the model to an image obtained by a relatively large motion. Perspective distortions can be seen in the table, the board, and the hanger at the upper right.

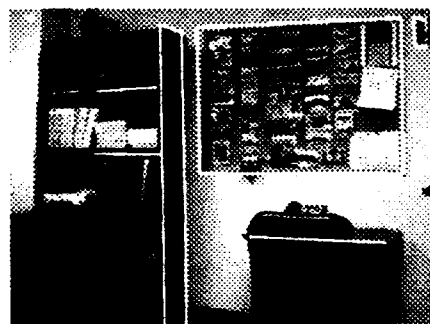
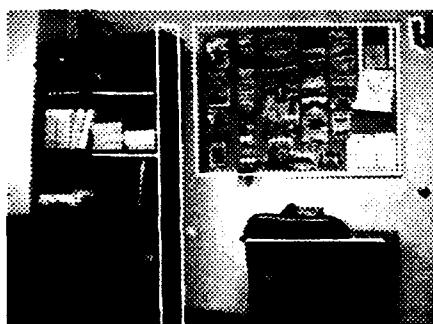


Figure 7: The results of applying the iterative process to reduce perspective distortions after one (left) and three (right) iterations.

- [10] M. J. Mataric. Environment learning using a distributed representation. *Proc. Int. Conf. on Rob. and Auto.*, Cincinnati, OH, 1990.
- [11] R. N. Nelson. Visual homing using an associative memory. *DARPA Image Understanding Workshop*, 245-262, 1989.
- [12] K. Onoguchi, M. Watanabe, Y. Okamoto, Y. Kuno, and H. Asada. A visual navigation system using a multi information local map. *Proc. Int. Conf. on Robotics and Automation*, Cincinnati, OH, pp. 767-774, 1990.
- [13] T. Poggio. 3D object recognition: on a result by Basri and Ullman. *TR 9005-03, IRST*, Povo, Italy, 1990.
- [14] E. Rivlin and R. Basri. Localization and positioning using combinations of model views. *TR-2926, Univ. of Maryland and AI-TR-1376, MIT*, 1992.
- [15] K. B. Sarachik. Visual navigation: constructing and utilizing simple maps of an indoor environment. *AI-TR-1113, MIT*, 1989.
- [16] D. W. Thompson and J. L. Mundy. Three dimensional model matching from an unconstrained viewpoint. *Proc. Int. Conf. on Rob. and Auto.*, Raleigh, NC, 208-220, 1987.
- [17] S. Ullman. Aligning pictorial descriptions: an approach to object recognition. *Cognition*, 32, 193-254, 1989.
- [18] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13, 992-1006, 1991.

Dynamic Camera Self-Calibration from Controlled Motion Sequences

Lisa Dron

M.I.T. Artificial Intelligence Laboratory
545 Technology Square, Cambridge, MA 02139

Abstract

In order to recover camera motion and 3-d structure from a sequence of images we must first relate points in the image plane to directions in space. This paper describes a least-squares algorithm for computing camera calibration from a series of motion sequences for which the translational direction of the camera is known. The method does not require special calibration objects or scene structure. It only requires the ability to move the camera in a given direction and to track features in the image as the camera moves. This method differs from other recently developed approaches in at least two respects. First, since it is a linear least-squares method, it can include information from many sequences to produce a robust estimate of the calibration matrix, which can be updated dynamically as new measurements are taken. Second, it uses the most general possible linear model for calibration, taking into account misalignment of the image sensor and optical axis, as well as rotation of the camera with respect to the external coordinate system. Experimental results from applying the algorithm to a set of real motion sequences with noisy correspondence data are given and analyzed.

1 Introduction

Before camera motion and 3-d structure can be deduced from a set of images, it is necessary to know the relation between points in the image plane and directions in 3-space. Methods for computing camera calibration differ mainly in the complexity of the camera model and the type of laboratory setup required. The accuracy of the calibration and the complexity of the model needed are determined by the application. For 3-d metrology, it is necessary to use as complete a model as possible which includes the nonlinear distortions due to imperfections and misalignments in the optical system. The basic method for determining the linear perspective parameters and nonlinear radial lens distortion using a carefully designed calibration pattern was developed by Tsai [Tsai, 1986], and later refined by Lenz and Tsai [Lenz and Tsai, 1988]. Since then the method has been improved several times by expanding the number of distortion parameters in the model and using nonlinear optimization techniques [Weng *et al.*, 1992].

For computing camera motion and judging relative distances as needed for navigation, the precision of the

full nonlinear model is not usually necessary. Since camera motion is estimated by measurements taken over the entire image, the effects of distortion are not as important unless the lens distortion is particularly severe. The previously cited methods require tedious measurements and carefully controlled setups. It is certainly desirable to avoid the use of calibration patterns if this is possible; and as a practical matter, it may be necessary to do so if the camera is placed in an environment where the scene cannot be controlled.

Recently, methods for computing the linear pinhole-model parameters of perspective projection by tracking features in the image, without using special patterns, have been developed by Faugeras *et al.* [Faugeras *et al.*, 1992] and Hartley [Hartley, 1992]. In the second method, the effective focal lengths and relative positions of two different cameras are computed from two images using an initial guess for the locations of their principle points. The method developed by Faugeras *et al.* applies theories from algebraic projective geometry to compute all of the pinhole model parameters from three image sequences.

The algorithm for camera self-calibration presented in this paper is similar to these methods in that it requires the ability to track features between images as the camera moves and does not need special calibration patterns. It differs from the previous methods, however, in that it is a linear least squares algorithm and can include information from many motion sequences to produce a robust estimate which can be updated dynamically as new measurements are taken. Furthermore it uses the most general linear camera model possible, taking into account misalignment of the image sensor and optical axis, as well as the rotation of the camera with respect to the motion stage coordinate system.

Unlike the other methods, it *does* require that the translational direction of the camera be known for each sequence. It is not necessary to know the rotation; although the estimation is simplified if it is made to be zero. The benefit gained in return for having to control the camera motion, is a less stringent requirement on the accuracy of the point correspondences than needed otherwise. For example, Faugeras *et al.* report that errors of 1 pixel in the location of the correspondences can significantly affect their results. Such precision is not achievable from block-matching techniques which can be

implemented in hardware. In many situations, it is much easier to move the camera in a known direction than to attain such high accuracy in the correspondences.

2 Basic Equations and Definitions

The least-squares algorithm for computing the camera calibration from a set of controlled motion sequences is presented in Section 3. In this section I define the notation which will be used later and rederive the fundamental equations describing the apparent motion of points in the image plane given the motion of the camera with respect to an external coordinate system.

2.1 The camera calibration matrix

Let $\mathbf{w} = (x, y, z)^T$ represent a point in the world with respect to the origin of the camera coordinate system. We are interested in relating the projection of \mathbf{w} onto the plane $(0, 0, f)^T$ to the coordinates of a point on a digitized pixel grid. Let $\tilde{\mathbf{w}} = (x/z, y/z, 1)^T$ represent the 2-dimensional homogeneous coordinates of \mathbf{w} . Two world points \mathbf{w}_i and \mathbf{w}_j are projectively equivalent on $(0, 0, f)^T$ if and only if $\tilde{\mathbf{w}}_i = \tilde{\mathbf{w}}_j$.

Image plane coordinates, $\mathbf{m} = (m_x, m_y, 1)^T$ are defined on a rectangular coordinate system such that (m_x, m_y) represents the coordinates of a point in the image plane with the centers of each pixel located at integer values of m_x and m_y . If we ignore the nonlinear radial and tangential distortions caused by imperfections in the lens and assume perfect perspective projection, the image plane coordinates are related to 2-d homogeneous world coordinates by a linear transformation matrix \mathbf{K}_c

$$\mathbf{m} = \mathbf{K}_c \tilde{\mathbf{w}} \quad (1)$$

Under certain conditions \mathbf{K}_c takes on a special form. If the axes of the pixel grid are exactly orthogonal, and if the image plane is exactly perpendicular to the optical axis, \mathbf{K}_c is upper triangular and has the form

$$\mathbf{K}_c = \begin{pmatrix} f/s_x & 0 & x_0 \\ 0 & f/s_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \quad (2)$$

where f is the effective focal length, s_x and s_y are the spacings between pixel centers along the orthogonal x - and y -axes, and (x_0, y_0) is the location of the principal point, in image plane coordinates, where the optical axis pierces the image plane.

In practice, the conditions for \mathbf{K}_c to have the form of (2) will not be met exactly. Synchronization error or clock skew in the frame capture hardware can cause the grid axes to be slightly non-orthogonal. Since commercial CCD cameras are not assembled with strict alignment tolerances, it is also unlikely that the image sensor will be exactly perpendicular to the optical axis. Taking these effects into consideration, a more general form for \mathbf{K}_c can be written as

$$\mathbf{K}_c = \mathbf{A}_s \begin{pmatrix} f/s_x & -f/(s_x \tan \theta) & x_0 \\ 0 & f/(s_y \sin \theta) & y_0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3)$$

where \mathbf{A}_s is the affine transformation between image plane coordinates on the rotated and unrotated image sensor, and θ is the angle between the grid axes.

2.2 Relative motion and the coplanarity constraint

Suppose we have two images taken by the same camera at two different positions, which we will refer to as *right* and *left*. Let $\{\mathbf{w}_{ri}, \mathbf{w}_{li}\}$ denote world coordinates with respect to the right and left coordinate systems of a set of fixed points in the environment, $\mathbf{w}_{ri} = (x_r, y_r, z_r)^T$ and $\mathbf{w}_{li} = (x_l, y_l, z_l)^T$. Assuming rigid motion of the camera with respect to the stationary environment, \mathbf{w}_{ri} and \mathbf{w}_{li} are related by

$$\mathbf{w}_{ri} = \mathbf{R}_c \mathbf{w}_{li} + \mathbf{t}_c \quad (4)$$

where \mathbf{R}_c denotes an orthonormal rotation matrix, and \mathbf{t}_c is the translation vector that connects the origins of the two systems.

In terms of homogeneous coordinates, $\tilde{\mathbf{w}}_r = (x_r/z_r, y_r/z_r, 1)^T$ and $\tilde{\mathbf{w}}_l = (x_l/z_l, y_l/z_l, 1)^T$, equation (4) can be rewritten as

$$z_r \tilde{\mathbf{w}}_r = z_l \mathbf{R}_c \tilde{\mathbf{w}}_l + \mathbf{t}_c \quad (5)$$

and hence,

$$z_r \mathbf{K}_c^{-1} \mathbf{m}_{ri} = z_l \mathbf{R}_c \mathbf{K}_c^{-1} \mathbf{m}_{li} + \mathbf{t}_c \quad (6)$$

A necessary condition for the vectors \mathbf{w}_{ri} and \mathbf{w}_{li} to intersect at the world point i is that they be coplanar with \mathbf{t}_c , or equivalently, that the triple product of the ray directions vanish

$$\mathbf{R}_c \tilde{\mathbf{w}}_{li} \cdot (\mathbf{t}_c \times \tilde{\mathbf{w}}_{ri}) = 0 \quad (7)$$

z_r and z_l drop out of the above expression since they are merely scale factors.

Equation (7) is known as the *coplanarity constraint* and is the basis of all methods for computing relative camera motion given a set of matched vectors $\{\mathbf{w}_{ri}, \mathbf{w}_{li}\}$. A similar constraint can be written in terms of image coordinates by writing (6) as

$$\begin{aligned} z_r \mathbf{m}_{ri} &= z_l \mathbf{K}_c \mathbf{R}_c \mathbf{K}_c^{-1} \mathbf{m}_{li} + \mathbf{K}_c \mathbf{t}_c \\ &= z_l \mathbf{U}_c \mathbf{m}_{li} + \mathbf{v}_c \end{aligned} \quad (8)$$

with

$$\mathbf{U}_c \equiv \mathbf{K}_c \mathbf{R}_c \mathbf{K}_c^{-1} \quad (9)$$

$$\mathbf{v}_c \equiv \mathbf{K}_c \mathbf{t}_c \quad (10)$$

Equation (10) is the key equation which will be used in the least-squares algorithm of Section 3. As with (5), we obtain

$$\mathbf{U}_c \mathbf{m}_{li} \cdot (\mathbf{v}_c \times \mathbf{m}_{ri}) = 0 \quad (11)$$

2.3 Motion stage orientation

In order to generate controlled motion sequences, the camera must be mounted on some device which is able to move in a precise manner. In practice, it is not feasible to exactly align the camera coordinate system with that of the motion stage, and hence we must also find the transformation which relates the motion of the stage to that of the camera.

Let $(\mathbf{R}_o, \mathbf{t}_o)$ denote the rotation and translation of the camera coordinate system with respect to the motion

stage coordinate system such that if $\mathbf{w}_c = (x_c, y_c, z_c)^T$ are the coordinates of a world point in the camera system and $\mathbf{w}_m = (x_m, y_m, z_m)^T$ are the coordinates of the same point with respect to the motion stage then

$$\mathbf{w}_c = \mathbf{R}_o \mathbf{w}_m + \mathbf{t}_o \quad (12)$$

Suppose the stage executes a motion described by a rotation and translation $(\mathbf{R}_m, \mathbf{t}_m)$ of its coordinate center. Let \mathbf{w}_{rm} and \mathbf{w}_{lm} denote the coordinates of the same world point with respect to the stage before and after the motion and let \mathbf{w}_{rc} and \mathbf{w}_{lc} denote the same points in the right and left camera systems. We have then

$$\mathbf{w}_{rm} = \mathbf{R}_m \mathbf{w}_{lm} + \mathbf{t}_m \quad (13)$$

and, combining equations (12) and (13), we obtain after some algebraic manipulation

$$\mathbf{w}_{rc} = \mathbf{R}_o \mathbf{R}_m \mathbf{R}_o^T \mathbf{w}_{lc} + \mathbf{R}_o \mathbf{t}_m + (\mathbf{I} - \mathbf{R}_o \mathbf{R}_m \mathbf{R}_o^T) \mathbf{t}_o \quad (14)$$

Comparing (14) with (4), we see that

$$\mathbf{R}_c = \mathbf{R}_o \mathbf{R}_m \mathbf{R}_o^T \quad (15)$$

$$\mathbf{t}_c = \mathbf{R}_o \mathbf{t}_m + (\mathbf{I} - \mathbf{R}_o \mathbf{R}_m \mathbf{R}_o^T) \mathbf{t}_o \quad (16)$$

and we can use (9) and (10) to determine that

$$\mathbf{U}_c \equiv \mathbf{K}_c \mathbf{R}_c \mathbf{K}_c^{-1} = \mathbf{K}_c \mathbf{R}_o \mathbf{R}_m \mathbf{R}_o^T \mathbf{K}_c^{-1} \quad (17)$$

and

$$\mathbf{v}_c \equiv \mathbf{K}_c \mathbf{t}_c = \mathbf{K}_c \mathbf{R}_o \mathbf{t}_m + \mathbf{K}_c (\mathbf{I} - \mathbf{R}_o \mathbf{R}_m \mathbf{R}_o^T) \mathbf{t}_o \quad (18)$$

Replacing \mathbf{K}_c by $\mathbf{K}_m \equiv \mathbf{K}_c \mathbf{R}_o$, we obtain finally

$$\mathbf{U}_c = \mathbf{K}_m \mathbf{R}_m \mathbf{K}_m^{-1} \quad (19)$$

$$\mathbf{v}_c = \mathbf{K}_m \mathbf{t}_m + \mathbf{K}_m (\mathbf{I} - \mathbf{R}_m) \mathbf{R}_o^T \mathbf{t}_o \quad (20)$$

which, except for the term multiplying \mathbf{t}_o , gives the same set of equations as (9) and (10) in terms of the rotation and translation of the motion stage, \mathbf{R}_m and \mathbf{t}_m , and the matrix \mathbf{K}_m .

There are two ways in which the \mathbf{t}_o term in the above expression can be effectively eliminated. The first is by making $\mathbf{R}_m = \mathbf{I}$, or at least $\mathbf{R}_m \approx \mathbf{I}$, which is usually possible since we control \mathbf{R}_m . The second is by making $\|\mathbf{t}_m\| \gg \|\mathbf{t}_o\|$. In either case, we arrive at an equation of the form (10) to be solved for \mathbf{K}_m instead of \mathbf{K}_c . Once \mathbf{K}_m is found, other procedures, which will not be discussed in this paper, may be applied if desired to estimate $\mathbf{R}_o^T \mathbf{t}_o$ and complete the camera-motion stage calibration.

3 Computing the Calibration Matrix

The algorithm for computing the calibration matrix \mathbf{K}_m is based on solving a set of equations of the form

$$\mathbf{v}_{ck} = \mathbf{K}_m \mathbf{t}_{mk} \quad (21)$$

given M motion sequences (pairs of images) for which the translation directions $\hat{\mathbf{t}}_{mk}$, $k = 1, \dots, M$, are known. Equation (21) is the same as (20) in which it is assumed that \mathbf{t}_o can be ignored, either by making $\mathbf{R}_m = \mathbf{I}$ or $\|\mathbf{t}_{mk}\| \gg \|\mathbf{t}_o\|$.

As explained in Section 4, it is numerically difficult to obtain a good estimate directly from the image data for $\mathbf{K}_m = \mathbf{K}_c \mathbf{R}_o$, with \mathbf{K}_c of the form (3), since this matrix is very poorly conditioned. Instead, we start with an estimate, \mathbf{K}_s , and look for the the best update matrix \mathbf{K}_u such that

$$\mathbf{K}_m = \mathbf{K}_s \mathbf{K}_u \quad (22)$$

The estimate \mathbf{K}_s can be derived either from the manufacturer's data on the camera and associated frame capture hardware, or by iteratively solving (22) and replacing \mathbf{K}_s with the value computed for \mathbf{K}_m on the previous iteration. We thus define

$$\mathbf{U}_{sk} \equiv \mathbf{K}_s^{-1} \mathbf{U}_{ck} \mathbf{K}_s \quad (23)$$

$$\mathbf{v}_{sk} \equiv \mathbf{K}_s^{-1} \mathbf{v}_{ck} \quad (24)$$

$$\mathbf{p} \equiv \mathbf{K}_s^{-1} \mathbf{m} \quad (25)$$

such that from (19) and (20)

$$\mathbf{U}_{sk} = \mathbf{K}_u^{-1} \mathbf{R}_{mk} \mathbf{K}_u \quad (26)$$

$$\mathbf{v}_{sk} = \mathbf{K}_u \mathbf{t}_{mk} \quad (27)$$

and

$$\mathbf{p} = \mathbf{K}_u \tilde{\mathbf{w}} \quad (28)$$

The algorithm for computing \mathbf{K}_u consists of first determining the uncalibrated translation directions $\hat{\mathbf{v}}_{sk}$ for each sequence, $k = 1, \dots, M$. Since only the direction of translation is recovered from the relative motion equations, the equations (27) cannot be solved directly. The next step is therefore to compute the scale factors $\alpha_k = \|\mathbf{v}_{sk}\|/\|\mathbf{t}_{mk}\|$ such that $\alpha_k \hat{\mathbf{v}}_{sk} = \mathbf{K}_u \hat{\mathbf{t}}_{mk}$. After determining the α_k we can then solve for \mathbf{K}_u .

Each of these steps will be described in more detail in the remainder of this section. To simplify the notation, since only unit vectors are used in the following derivations, we will define

$$\hat{\mathbf{t}} \equiv \hat{\mathbf{t}}_m \quad (29)$$

$$\hat{\mathbf{v}} \equiv \hat{\mathbf{v}}_s \quad (30)$$

3.1 Computing the uncalibrated translation direction

For each sequence, $k = 1, \dots, M$ we first determine a set of correspondence points between the two images. The fundamental equation for computing relative camera motion is the coplanarity constraint which in terms of \mathbf{U}_s , \mathbf{v} , \mathbf{p}_{li} , and \mathbf{p}_{ri} is given by

$$\mathbf{U}_s \mathbf{p}_{li} \cdot (\mathbf{v} \times \mathbf{p}_{ri}) = 0 \quad (31)$$

These equations can be written in matrix form by replacing the cross product by an equivalent matrix multiplication. Let \mathbf{V}_x denote the skew-symmetric matrix

$$\mathbf{V}_x = \begin{pmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{pmatrix} \quad (32)$$

where (v_x, v_y, v_z) are the components of \mathbf{v} . Clearly $\mathbf{V}_x \mathbf{p}_{ri} = \mathbf{v} \times \mathbf{p}_{ri}$ and (31) can be written as

$$\mathbf{p}_{li}^T \mathbf{U}_s^T \mathbf{V}_x \mathbf{p}_{ri} = 0 \quad (33)$$

The above formulation is identical to that derived by Longuet-Higgins [Longuet-Higgins, 1981] who proposed a linear method to compute the product matrix $Q = U_s^T V_x$ to within an arbitrary scale factor from 8 correspondence points. Once Q is known, v is identified as the eigenvector of $Q^T Q = vv^T - v^T v I$ with the smallest eigenvalue.

The Longuet-Higgins method, however, is numerically unstable. Since it does not explicitly enforce dependencies between the elements of Q and does not take into account any error in the data, it requires knowing the locations of the correspondence points with high accuracy. If there is error, as there usually is, the coplanarity condition will not hold exactly for all pairs of corresponding points. Equation (31) should instead be written as

$$U_s p_{li} \cdot (v \times p_{ri}) = e_i \quad (34)$$

and solved by a least-squares procedure which minimizes $\sum e_i^2$.

If the motion sequences are controlled such that $R_m = I$, then by (19) and (26) $U_s = I$, and equation (31) becomes

$$p_{li} \cdot (v \times p_{ri}) = v \cdot (p_{ri} \times p_{li}) = e_i \quad (35)$$

Given N correspondences and defining $y_i = p_{ri} \times p_{li}$

$$\sum_{i=1}^N e_i^2 = \sum_{i=1}^N v^T y_i y_i^T v \quad (36)$$

$$= v^T \left(\sum_{i=1}^N y_i y_i^T \right) v \quad (37)$$

$$= v^T Y v \quad (38)$$

The unit vector v which minimizes the sum of squared errors is the eigenvector of Y with the smallest eigenvalue.

The method of requiring $R_m = I$ so that v can be computed directly as an eigenvector of Y is the simplest procedure, but is not always practical. In many applications, it may be possible to translate the camera accurately in a known direction, but not to ensure $R_m = I$ with high precision. An example is a camera mounted on a vehicle which can move reliably from point to point, but has limited ability to align itself exactly in the direction of the previous position. In general it is best to estimate both U_s and v together. If in fact $U_s = I$, nothing is lost.

Weng *et al.* [Weng *et al.*, 1989] proposed an improved version of the Longuet-Higgins method which does minimize the sum of squared errors by using more than 8 correspondences and solving a 9×9 eigenvalue-eigenvector problem instead of computing the exact solution to the homogeneous linear equations. This method, however, still does not enforce dependencies between elements of Q , and, since it involves computing the eigenvector corresponding to the smallest eigenvalue of a relatively large matrix, it has inherent numerical difficulties. When the correspondence data are noisy, the results from this method have been found to be less reliable than those from other procedures.

The method which in testing the calibration algorithm was found to yield the best results, is to minimize

$$\sum_{i=1}^N e_i^2 = \sum_{i=1}^N (U_s p_{li} \cdot (v \times p_{ri}))^2 \quad (39)$$

explicitly assuming $U_s = K_u R_m K_u^{-1}$ is an orthonormal rotation matrix. Horn [Horn, 1991] developed an iterative algorithm to compute relative motion by directly solving the nonlinear minimization problem while enforcing the orthonormality of U_s . A revised version of Horn's algorithm, which was modified for more efficient implementation in hardware [Dron, 1992], was used for the results of Section 5.

If $R_m \neq I$ and the initial estimate K_s is far from K_m the assumption that U_s is orthonormal will not be very good. Nonetheless, as long as the relative motion algorithm can determine a well-defined solution to (39) for a sufficient number of motion sequences, the least-squares algorithm for computing the update K_u to the calibration matrix will find a solution that brings the new estimate closer to the correct K_m . As $K_s \rightarrow K_m$, $K_u \rightarrow I$, and the assumption of orthonormality for U_s will improve. A detailed analysis has not been performed to prove that an iterative procedure defined in this manner starting from an arbitrary K_s will always converge to the correct K_m . However, in the tests which have been conducted, the procedure has in fact converged to the same solution starting from very different initial values for K_s .

3.2 Estimating the α_k

Having computed the uncalibrated translation directions, v_k , we now have from (27) a set of equations

$$\alpha_k v_k = K_u t_k \quad (40)$$

for $k = 1, \dots, M$, where α_k is an unknown scale factor.

To compute a least-squares estimate for the factors α_k we choose three non-coplanar vectors, (t_1, t_2, t_3) , out of the M known translations to serve as a basis. For best numerical stability, it is desirable that (t_1, t_2, t_3) be mutually orthogonal; however the algorithm only requires that they span \mathbf{R}^3 . Let T be the matrix whose columns are the basis vectors

$$T = (t_1 \ t_2 \ t_3) \quad (41)$$

and let V be the matrix whose columns are the corresponding computed displacement vectors

$$V = (v_1 \ v_2 \ v_3) \quad (42)$$

From (40)

$$V \begin{pmatrix} \alpha_1 & 0 & 0 \\ 0 & \alpha_2 & 0 \\ 0 & 0 & \alpha_3 \end{pmatrix} = K_u T \quad (43)$$

Since T spans \mathbf{R}^3 , we can express the other translation vectors, t_k , $k = 4, \dots, M$, as linear combinations of (t_1, t_2, t_3)

$$\begin{aligned} t_k &= c_{k1} t_1 + c_{k2} t_2 + c_{k3} t_3 \\ &= T c_k \end{aligned} \quad (44)$$

with $c_k \equiv (c_{k1}, c_{k2}, c_{k3})^T$. Hence

$$\begin{aligned}\alpha_k v_k &= K_u t_k \\ &= K_u T c_k\end{aligned}\quad (45)$$

Let C_k be the diagonal matrix, $\text{diag}(c_{k1}, c_{k2}, c_{k3})$ formed from the elements of c_k and let $\alpha \equiv (\alpha_1, \alpha_2, \alpha_3)^T$, then from (43) and (45)

$$\alpha_k v_k = V C_k \alpha \quad (46)$$

Since v_k is a unit vector,

$$\alpha_k = v_k^T V C_k \alpha \quad (47)$$

and by substituting (46) into (47) we obtain

$$v_k v_k^T V C_k \alpha = V C_k \alpha \quad (48)$$

or,

$$(I - v_k v_k^T) V C_k \alpha = 0 \quad (49)$$

The fact that the above expression is homogeneous reflects the fact that K_u is known only to within an arbitrary scale factor, and hence that we can only solve for the relative proportions of the α_k . To fix the scale, we impose the condition $\|\alpha\| = 1$.

Since the computed estimates of v_k will necessarily contain some error, we should write (49) as

$$(I - v_k v_k^T) V C_k \alpha = e_k \quad (50)$$

where e_k is the error vector whose squared magnitude is

$$e_k^T e_k = \alpha^T C_k V^T (I - v_k v_k^T) V C_k \alpha \quad (51)$$

The total error for all of the sequences is therefore minimized by taking α to be the eigenvector corresponding to the smallest eigenvalue of

$$W \equiv \sum_{i=1}^M C_i V^T (I - v_i v_i^T) V C_i \quad (52)$$

In order to obtain a nontrivial, unique solution for α , we must have at least four motion sequences, including the basis, T , for which no three translations, t_k , are coplanar. If we have more than four sequences, the condition for obtaining a nontrivial, unique solution is that we can construct, though linear combinations of the t_k , at least four vectors, no three of which are coplanar. It should be noted that this condition is equivalent to that for being able to form a projective basis for \mathbb{R}^3 [Semple and Kneebone, 1952].

3.3 Computing K_u

Given the least-squares estimate for α , we can compute α_k for each sequence from equation (47). We now have from (40)

$$K_u t_k = \alpha_k v_k \quad (53)$$

in which the only unknown is the matrix K_u . Again, we consider the error in the estimated v_k and write

$$e_k = K_u t_k - \alpha_k v_k \quad (54)$$

As before, we seek to minimize the total error

$$\sum_{i=1}^M e_i^T e_i = \sum_{i=1}^M (K_u t_i - \alpha_i v_i)^T (K_u t_i - \alpha_i v_i) \quad (55)$$

The matrix K_u which minimizes the sum of squared error magnitudes satisfies

$$\frac{\partial \sum_{i=1}^M e_i^T e_i}{\partial K_u} = 0 = 2 \sum_{i=1}^M (K_u t_i t_i^T - \alpha_i v_i t_i^T) \quad (56)$$

from which we find that

$$K_u = \left(\sum_{i=1}^M \alpha_i v_i t_i^T \right) \left(\sum_{i=1}^M t_i t_i^T \right)^{-1} \quad (57)$$

This expression can be simplified by substituting equation (47) for α_k , and defining

$$\begin{aligned}f_k &\equiv C_k V^T v_k \\ D &\equiv \sum_{i=1}^M t_i t_i^T\end{aligned}\quad (58)$$

so that

$$K_u = \left(\sum_{i=1}^M (\alpha_i^T f_i) v_i t_i^T \right) D^{-1} \quad (59)$$

3.4 Including new measurements

A feature of a linear least-squares algorithms is that it is possible to improve a given estimate for K_u by performing subsequent measurements using a recursive update procedure, without having to recompute everything from the beginning.

Suppose we have $n-1$ measurements and have computed

$$W^{(n-1)} = \sum_{i=1}^{n-1} C_i V^T (I - v_i v_i^T) V C_i \quad (60)$$

We subsequently obtain the n th measurement and compute

$$W^{(n)} = W^{(n-1)} + C_n V^T (I - v_n v_n^T) V C_n \quad (61)$$

from which we obtain a new estimate $\alpha^{(n)}$.

Now define

$$F_1^{(n-1)} = \sum_{i=1}^{n-1} f_{i1} v_i t_i^T \quad (62)$$

$$F_2^{(n-1)} = \sum_{i=1}^{n-1} f_{i2} v_i t_i^T \quad (63)$$

$$F_3^{(n-1)} = \sum_{i=1}^{n-1} f_{i3} v_i t_i^T \quad (64)$$

where (f_{i1}, f_{i2}, f_{i3}) are the components of f_i . The matrices $F_1^{(n-1)}$, $F_2^{(n-1)}$ and $F_3^{(n-1)}$ are updated in the obvious way, and we can use a matrix identity to compute

$$\begin{aligned}D^{(n)-1} &= (D^{(n-1)} + t_n t_n^T)^{-1} \\ &= D^{(n-1)-1} - \frac{D^{(n-1)-1} t_n t_n^T D^{(n-1)-1}}{1 + t_n^T D^{(n-1)-1} t_n}\end{aligned}\quad (65)$$

$K_u^{(n)}$ can then be computed directly from

$$K_u^{(n)} = (\alpha_1^{(n)} F_1^{(n)} + \alpha_2^{(n)} F_2^{(n)} + \alpha_3^{(n)} F_3^{(n)}) D^{(n)-1} \quad (66)$$

4 Numerical Stability and Parameter Estimation

Before applying the algorithm to real data, we should first address the issues of numerical stability and analyze the effects of error on the estimation of the different components of the calibration matrix.

4.1 Numerical stability

The structure of K_c , given by equation (2), guarantees numerical problems. Even though we compute K_m , which in general differs from the form of (2) by the affine and rotational transformations, A_s and R_o , the weakness of the underlying structure remains.

Typical commercial CCD image arrays are several hundred pixels wide in both horizontal and vertical dimensions. The effective focal length of the lens, which is comparable to the sensor dimensions, is therefore several hundred times longer than the spacing between the rows of sensing elements, and as a result, f/s_x , f/s_y , x_0 , and y_0 are all $\gg 1$. Suppose K_c is of the form (2). If we use image plane coordinates directly then

$$v_c = K_c t_c = \begin{pmatrix} t_x f/s_x + t_z x_0 \\ t_y f/s_y + t_z y_0 \\ t_z \end{pmatrix} \quad (67)$$

and the z , or third, component of the uncalibrated translation directions will therefore always be much smaller than the other two. Aside from the fact that it is difficult numerically just to estimate such small z components in the computed vectors v_k , the matrix V , which is central to the algorithm, will be nearly singular. Consequently, it is not feasible to estimate K_m in one step directly from the image data. As previously stated, we must instead start with an initial estimate K_s and compute the best update matrix K_u such that

$$K_m = K_s K_u \quad (68)$$

If K_s is not close to the actual K_m , several update iterations may be required before the estimate stabilizes. Although the least-squares algorithm for computing K_u is linear, the procedure for estimating the directions v_k by approximating U_s as an orthonormal matrix is not.

4.2 Parameter Sensitivity

To analyze the effect of error on estimating the different components of K_m , it is easiest to look at how errors in K_m affect the computed vectors v_k . First consider the effects of error in the scale factors and the principal point, assuming $A_s = R_o = I$ and that K_m has the form of equation (2). Let

$$\begin{aligned} K_m &= \begin{pmatrix} e_x f/s_x & 0 & e_x(x_0 + \delta_x) \\ 0 & e_y f/s_y & e_y(y_0 + \delta_y) \\ 0 & 0 & 1 \end{pmatrix} \quad (69) \\ &= \begin{pmatrix} e_x & 0 & e_x \delta_x \\ 0 & e_y & e_y \delta_y \\ 0 & 0 & 1 \end{pmatrix} K_m^* \\ &= K_{err} K_m^* \end{aligned}$$

From (40)

$$\alpha_k v_k = K_m t_k = K_{err} K_m^* t_k = \alpha_k^* K_{err} v_k^* \quad (70)$$

where K_m^* , α_k^* and v_k^* represent the error-free values of K_m , α_k and v_k , respectively.

Clearly, errors in scaling, represented by $e_x \neq 1$ and $e_y \neq 1$ will significantly affect the computed v_k . Suppose then that $e_x = e_y = 1$ and consider now the effect of error in x_0 , y_0 represented by δ_x , δ_y . We have

$$\alpha_k v_k = \alpha_k^* v_k^* + \alpha_k^* v_{kz}^* \begin{pmatrix} \delta_x \\ \delta_y \\ 0 \end{pmatrix} \quad (71)$$

By the same argument as before, v_{kz}^* will always be much smaller than the other components of v_k^* . From (71) we can see that errors in the principal point will have little effect on computing v_k ; and they will have no effect unless the z component of t_k is significantly different from zero.

These arguments can be turned around to conclude that we should be able to estimate the diagonal scale factors reasonably well, but that much greater precision in computing v_k is required in order to obtain an accurate estimate of the principal point.

Now consider the effects of the affine transformation A_s and the offset rotation R_o . A simple example illustrates the problems caused by these matrices for estimating the other parameters. Suppose $A_s = I$ and R_o represents a small rotation of θ about the y -axis, so that we can express K_m as

$$K_m = \begin{pmatrix} f/s_x - x_0\theta & 0 & f/s_x\theta + x_0 \\ -f/s_y\theta & f/s_y & y_0 \\ -\theta & 0 & 1 \end{pmatrix} \quad (72)$$

Since θ is small, the elements in the lower triangle of K_m are also small relative to the other elements of K_m , and hence will be difficult to estimate accurately. They can, however, significantly affect the estimates of the principal point and diagonal scale factors, if they are ignored. With $f/s_x = 567$, and $x_0 = 378$ —which are the values supplied by the manufacturer of the camera and lens used in the experiments—a rotation of 1° would cause a 10 pixel difference in the estimate of x_0 and a difference of 7 in the estimate of f/s_x .

In conclusion, we see that it is difficult to accurately estimate any of the individual parameters that compose K_m with great accuracy. The goal of the method presented in this paper, however, is not to recover the individual parameters but rather K_m itself. We do expect, that the matrix found by the algorithm will be approximately upper triangular of the form (2), and the fact that it is can be used as a partial confirmation that the result is reasonable. The real test of correctness, however, will be how well the computed motion given K_m fits the known motion of the camera and how closely subsequent motion sequences, not included in the computation of K_m , can be predicted.

5 Experimental Results

The algorithm was tested on a Cohu digital camera rigidly mounted on a movable carriage which could be translated along a fixed rail. The carriage assembly could be rotated on both the vertical and horizontal axes so that the camera could be oriented in any direction as

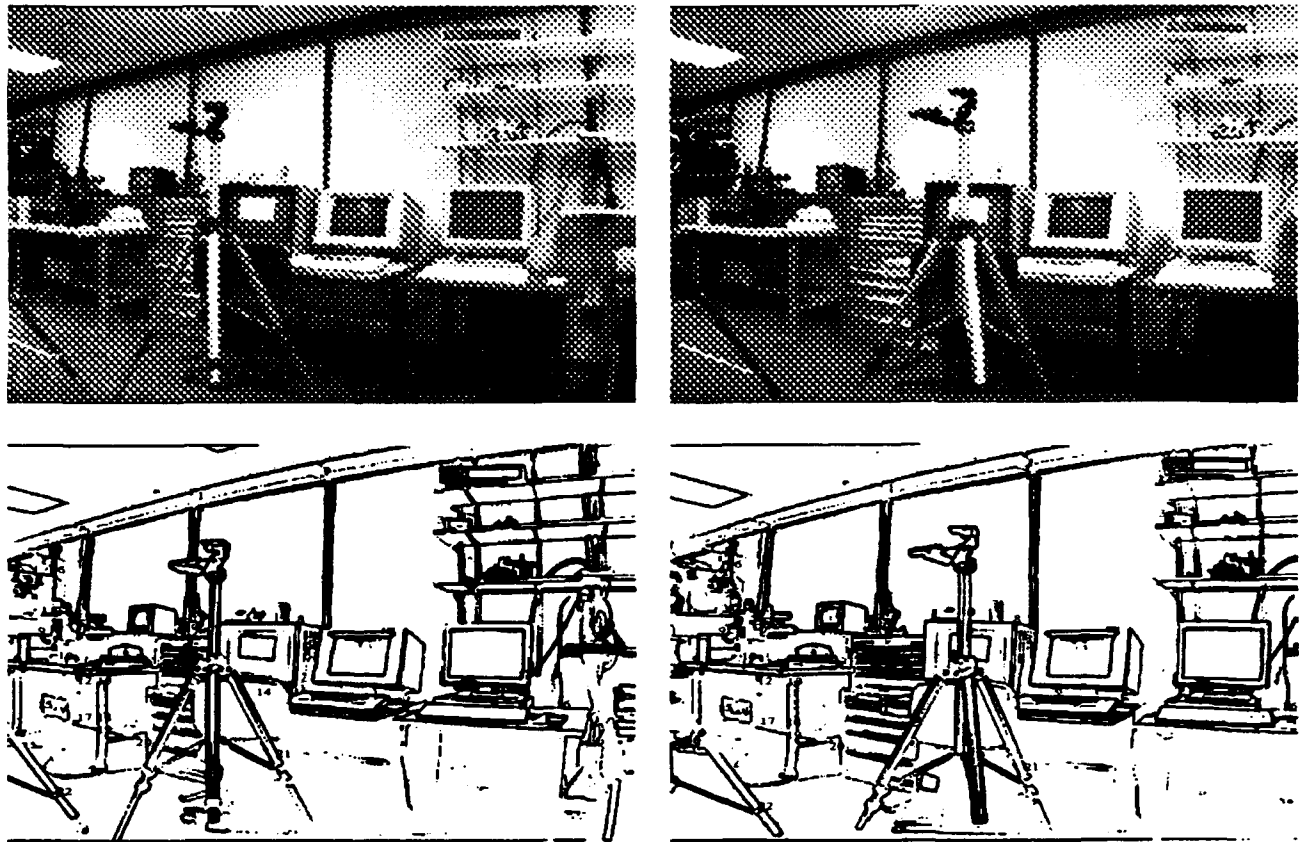


Figure 1: Motion sequence #3, $t_m = (.866, 0, -0.5)$. Top: original images. Bottom: edge maps with numbered matched points.

the assembly was moved along the rail. Positional accuracy was better than $.1^\circ$ on each axis.

Digitized image data was read directly from the camera over an Sbus interface to a Sparc workstation. The advantage of the digital camera is that each pixel corresponds exactly to a sensor location. Since there is no frame grabber in the path to resample and resize the data, the calibration matrix should be very close to that given by the manufacturer's data on the image sensor. According to the data sheets, the sensor is $6.4\text{mm} \times 4.8\text{mm}$ with 756 (horizontal) and 484 (vertical) pixels. A 4.8mm lens was used to give an approximately 80° field of view. This information was used to generate the initial calibration matrix given in table 3.

Twelve pairs of images were taken for the relative motions given in table 1. Not all of these were purely translational motion. Sequences #6 and #10 combined a y -axis rotation of 5° , and sequence #9 included a 5° rotation about z . Point correspondences were found using an improved version of the matching procedure described in [Dron, 1992]. With this method, the binary edge map of one of the images is divided into blocks which are individually shifted across the edge map from the other image in search of the offset which gives the best alignment. Only the most reliable matches are kept. For each pair of images in these sequences there were generally 20 to 30 correspondence points retained. Figure 1 shows

one of the images pairs (sequence #3) along with the binary edge maps upon which the correspondence points are marked. One can see from these images that there is a fair amount of nonlinear distortion in the periphery produced by imperfections in the small focal-length lens.

The improvements made to the matching procedure in [Dron, 1992] were to reject any match which did not have a well-localized peak and to use knowledge of the translation direction to shape the search window. Since the procedure compares 24×24 blocks, there is an inherent uncertainty of ~ 3 -4 pixels caused by defining the location of the correspondence to be at the centers of the blocks. However, no adjustments were made to the locations given by the matching procedure.

Uncalibrated translation directions were computed for each sequence from the point correspondences found by the matching procedure using the relative motion algorithm described in [Dron, 1992], which is a modified version of Horn's algorithm [Horn, 1991]. This algorithm produces a least-squares estimate of the translation direction assuming the matrix U , from (26) is orthonormal. However, since it is a nonlinear optimization problem, it can get stuck in local minima and produce erroneous results. Two steps were taken to ensure that the uncalibrated translation directions computed from the algorithm were reliable. First, the motion was estimated twice. After the first pass, the errors in the

	Motion Sequences											
	1	2	3	4	5	6	7	8	9	10	11	12
t_x	1.0000	0.0	0.0	0.7071	0.8660	0.5000	0.0	0.0	0.0	0.7071	0.6124	0.3536
t_y	0.0	1.0000	0.0	-0.7071	0.0	0.0	0.8660	0.5000	0.0	-0.7071	-0.6124	-0.3536
t_z	0.0	0.0	1.0000	0.0	-0.5000	-0.8660	0.5000	0.8660	1.0000	0.0	-0.5000	-0.8660

Table 1: Camera motions used to generate test image pairs

	Computed Motion											
	1	2	3	4	5	6	7	8	9	10	11	12
t_x	0.9993	-0.0071	-0.0010	0.7007	0.9081	0.4921	0.0137	-0.0029	-0.0107	0.6977	0.5564	0.2804
t_y	-0.0316	0.9994	0.0100	-0.7120	0.0500	0.0163	0.8564	0.4754	0.0131	-0.7071	-0.6760	-0.2808
t_z	-0.0209	0.0350	0.9999	-0.0467	-0.4159	-0.8704	0.5161	0.8798	0.9999	-0.1150	-0.4831	-0.9179
Δ_k	0.0004	0.0003	0.0000	0.0005	0.0028	0.0001	0.0002	0.0002	0.0001	0.0033	0.0019	0.0033

Table 2: Computed translation directions for test image pairs from calibration matrix of T_p^{-1} 4

coplanarity constraint (31) were computed, and every correspondence pair whose error was greater than one standard deviation above the mean squared error was removed from the list. The translation direction used in the calibration algorithm was the one computed from the remaining correspondence points with the least error.

The second step taken was to judge the reliability of the result from the condition number of the translation matrix. At each iteration of the relative motion algorithm, a matrix of the form of Y in equation (38) is computed, given the matrix U_s . The estimated translation v is the eigenvector corresponding to the smallest eigenvalue of this matrix in the final iteration. If the condition number of this matrix is small, the estimate of v is very sensitive to error, and the least eigenvalue reported may in fact correspond to a vector that is perpendicular to the correct translation direction. Hence any sequence for which the condition number was < 1000 was categorically rejected and was not used to compute the calibration matrix.

The results of applying the least-squares algorithm for camera calibration to the data from these sequences are given in Tables 4-7. Note that both K_m and K_m^{-1} as given in the tables have been normalized so that their (3,3) element is 1. Hence they are not exact inverses of each other.

The results of two experiments are shown. In the first, the calibration matrix given in Table 3, which was derived from the manufacturer's data, was used as a initial estimate. An update matrix was computed from sequences 1-8, and the result is given in Table 4. Estimated translation directions for all 12 motion sequences were then computed by the relative motion algorithm using this new matrix, and the results are given in Table 2. The prediction error, computed from $\Delta_k = (1 - v_k^T t_k)/2$, based on the actual motion t_k is also given at the bottom of the table. As expected, the error is smallest for vectors 1-8 which were used to compute the new calibration matrix. However, the errors for the four sequences, 8-12, which were not used are also reasonably small.

The second experiment consisted of deriving the calibration matrix using the iterative method described in

Section 4 without any prior knowledge of the camera system. Tables 5-7 show the results after iterations 1, 3, and 6, starting initially from $K_s = I$. By the 6th iteration the result is very close to that of Table 4 which was computed starting from the manufacturer's data.

6 Conclusions and Future Work

The least-squares method derived in this paper for computing camera calibration has been shown to work well for the real image sequences given in the last section. These results are very encouraging; however, additional tests should be conducted on different scenes and different cameras to confirm the general applicability of the method.

An eventual goal is to implement the automatic calibration procedure in hardware as part of a larger navigation system. This was a primary motivation for developing a method which could tolerate substantial error in the correspondences. Before doing so, however, a more extensive analysis should be performed to determine how much error in the data can be tolerated as well as to determine more precisely how nonlinear distortion in the lens affects the results.

Acknowledgements

I wish to acknowledge my fellowship sponsor, AT&T Bell Laboratories, who has generously supported me for the last three and a half years. This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124. Funding for the Analog VLSI for Machine Vision research project is provided by NSF Grant MIP-91-17724 and NSF and DARPA contracts MIP-88-14612.

Initial Calibration from Manufacturer's Data					
K_m			K_m^{-1}		
567	0.0	378	0.001764	0.0	-0.666667
0.0	484	242	0.0	0.002066	-0.500000
0.0	0.0	1.0	0.0	0.0	1.000000

Table 3: Initial calibration matrix derived from manufacturer's data on the image sensor.

Calibration Computed from Sequences 1-8 using Manufacturer's Data for K_s					
K_m			K_m^{-1}		
747.0	-20.3	379.3	0.001334	0.000047	-0.517834
23.8	638.8	249.1	-0.000037	0.001545	-0.370715
0.0	0.0	1.0	-0.000032	-0.000010	1.000000

Table 4: Calibration matrix computed from sequences 1-8 starting with matrix of Table 3 for K_s .

Calibration Computed from Sequences 1-8 after 1st iteration with $K_s = I$					
K_m			K_m^{-1}		
1503.5	22.1	405.1	0.000654	0.000034	-0.268187
-16.52	1120.0	97.4	0.000025	0.000769	-0.084876
0.3	0.2	1.0	-0.000191	-0.000168	1.000000

Table 5: Calibration matrix computed from sequences 1-8 after 1st iteration starting with $K_s = I$

Calibration Computed from Sequences 1-8 after 3rd iteration from $K_s = I$					
K_m			K_m^{-1}		
927.6	-8.0	313.9	0.001103	-0.000019	-0.341880
-111.0	632.5	217.7	0.000097	0.001717	-0.404238
-0.2	-0.1	1.0	0.000281	0.000101	1.000000

Table 6: Calibration matrix computed from sequences 1-8 after 3rd iteration starting from $K_s = I$

Calibration Computed from Sequences 1-8 after 6th iteration from $K_s = I$					
K_m			K_m^{-1}		
864.9	11.0	363.0	0.001146	-0.000004	-0.415092
17.6	646.7	240.6	-0.000022	0.001533	-0.360741
0.0	0.0	1.0	-0.000024	-0.000037	1.000000

Table 7: Calibration matrix computed from sequences 1-8 after 6th iteration starting from $K_s = I$

References

- [Dron, 1992] Lisa Dron. System-level design of specialized VLSI hardware for computing relative orientation. In *Proceedings IEEE Workshop on Applications of Computer Vision, November 30 - December 2, 1992*, pages 128-135, Palm Springs, CA, 1992.
- [Faugeras et al., 1992] Olivier D. Faugeras, Q.-T. Luong, and Steve J. Maybank. Camera self-calibration: Theory and experiments. In *Proceedings of the Second European Conference on Computer Vision*, pages 321-334, Santa Margherita Ligure, Italy, May 1992.
- [Hartley, 1992] Richard I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Proceedings of the Second European Conference on Computer Vision*, pages 579-587, Santa Margherita Ligure, Italy, May 1992.
- [Horn, 1991] Berthold K. P. Horn. Relative orientation revisited. *Journal of the Optical Society of America*, 8:1630-1638, October 1991.
- [Lenz and Tsai, 1988] Reimar K. Lenz and Roger Y. Tsai. Techniques for calibration of the scale factor and image center for high accuracy 3-d machine vision metrology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):713-720, September 1988.
- [Longuet-Higgins, 1981] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133-135, 1981.
- [Semple and Kneebone, 1952] J. G. Semple and G. T. Kneebone. *Algebraic Projective Geometry*. Clarendon Press, Oxford, 1952.
- [Tsai, 1986] Roger Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 364-374, Miami Beach, FL, June 1986.
- [Weng et al., 1989] Juyang Weng, Thomas S. Huang, and Narendra Ahuja. Motion and structure from two perspective views: Algorithms, error analysis, and error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):451-476, May 1989.
- [Weng et al., 1992] Juyang Weng, Paul Cohen, and Marc Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):965-980, October 1992.

Fast and Robust 3D Recognition by Alignment

T. D. Alter and W. Eric L. Grimson
MIT AI Lab & Dept. of EECS
Cambridge, Massachusetts 02139

Abstract

Alignment is a prevalent approach for recognizing three-dimensional objects in two-dimensional images. Current implementations handle errors that are inherent in images in ad hoc ways. These errors, however, can propagate and magnify through the alignment computations, such that the ad hoc approaches may not work. This paper gives a technique for tightly bounding the propagated error, which can be used to make the recognition robust while still being efficient.

Previous analyses of alignment have indicated that the approach is sensitive to false positives, even in moderately-cluttered scenes. But these analyses applied only to point features, whereas almost all alignment systems rely on extended features, such as line segments, for verifying the presence of a model in the image. We derive a new formula for the "selectivity" of a line feature. Then,

using the technique for computing error bounds, it is demonstrated experimentally that the use of line segments significantly reduces the expected false positive rate. The extent of the improvement is that an alignment system that correctly handles propagated error is expected to remain reliable even in substantially-cluttered scenes.

1 Introduction

Object recognition involves determining which of a set of known objects are in an image and where they are. Many systems use rigid objects, modeled by geometric *features* such as lines and points. Given a set of such object models, the task of model-based recognition is to find correspondences between model features and their projections in the image. To find large sets of correspondences, many approaches begin with minimal sets—i.e. sets with sufficient matches to transform a model to the image—and try to extend them. Backtracking search starts from each minimal set and repeatedly uses the current matches to constrain the search for an additional match, backtracking whenever an inconsistency is found (e.g., [6, 13, 14]). Transform clustering uses every minimal set to compute a model-to-image transformation, then counts how often each transformation is repeated (e.g., [4, 17, 7]). Alignment methods use each minimal set to transform all the model features into the image, then look near each predicted model feature for a matching image feature (e.g., [8, 3, 5, 15, 16]).

Here, we focus on alignment, analyzing the effects of uncertainty in the image features. To be

⁰A similar paper will appear in the Inter. Conf. on Computer Vision [May, 1993]. This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Army contract number DACA76-85-C-0010 and under Office of Naval Research contract N00014-85-K-0124. T. D. Alter was supported in part by an NDSEG Fellowship. WELG was supported in part by NSF contract number IRI-8900267.

robust to such uncertainty, an alignment system needs to know, for each predicted model feature, where in the image to search for a matching image feature. If the regions searched are smaller or larger than necessary, correct matches may be missed (*false positives*) and incorrect matches may be accepted (*false negatives*). Assuming "weak-perspective" (i.e. scaled orthographic) projection and a "bounded error" model of uncertainty in the image points, we give a method for computing the correct search regions for point and line features of a 3D model, given a set of 3 corresponding model and image points, since 3 is minimal [15].

Even if the correct search regions are known, there may still be false positives, i.e., image features appearing at random in the search regions, causing incorrect identifications of models in the image. Consequently the reliability of an alignment system depends largely on its false positive rate. We use the search regions to compute the probability of a false positive and then use the probability of a false positive to compute limits on the amount of scene clutter that alignment can handle. We also examine how much of the model must be matched to keep the probability of a false positive low.

The approach we consider uses points for generating minimal sets (hypotheses) and then verifies the hypotheses using extended features as well as points (as in [15]). Specifically, Section 2 describes a theory for analyzing the false-positive sensitivity of an alignment system that uses points or line features for verification. The main result is a formula for the "selectivity" of a line feature. Section 3 computes the propagated uncertainty regions for point features, which give the regions for line features (Section 2.1). We show that the regions for points can be computed quickly and accurately by fitting circles to their sampled boundaries, assuming a fast solution for the image position of a predicted model point given 3 corresponding model and image points (as in [1]). Section 4 uses the formulas for selectivity and false positives (Section 2) and the technique for computing propagated error bounds (Section 3) to estimate the improvement gained by using line features for verification, and to judge how sensitive alignment is to false positives.

Previous work has shown that the propagated uncertainty regions for 3D point features can be computed exactly for flat models [16]. For 3D solid

models, [11] uses numerically computed bounds on the parameters of the model-to-image transformation to estimate conservative bounds on the propagated regions. Analyses of false positive rates have been provided for recognition involving 2D models and 2D data for both point and line features [9, 10]. Using point features only, similar analyses have been applied to recognition involving 3D models and 2D data for flat models [12], and also for solid models using the bounds of [11].

2 Analyzing the False-Positive Sensitivity of Alignment

2.1 Propagated uncertainty regions

A match of three model and image points determines the image position of an unmatched model point up to a finite number of solutions [8, 15]. Any uncertainty in the locations of the matched image points propagates to uncertainty in the predicted position of an unmatched model point. Errors in the sensed or *nominal* locations of the image points are assumed to be bounded by circles of radius ϵ . Then, as the three image points move independently around their ϵ -circles, the fourth model point traces out a region of possible image locations. Any image point within ϵ of this region is a possible match for the model point.

For flat objects, the propagated uncertainty region for a model point is a disc centered at the nominal location, whose radius depends on the affine coordinates of the nominal location with respect to the basis triple [16]. For solid models, we assume that the uncertainty region can be bounded accurately with an "uncertainty circle" centered at the nominal point. Section 3 demonstrates that this assumption generally is valid.

We can use this result to bound the uncertainty in predicted line segments. We assume that the orientation of an image line segment is constrained such that its endpoints lie within ϵ circles. Then, for each model line segment we calculate the uncertainty circles for its endpoints. Next, we find candidates for each model segment by gathering all image line segments that lie entirely within the uncertainty region formed by the uncertainty circles and their common outer tangents (Fig. 1) and whose extensions intersect both of the uncertainty circles.

2.2 Selectivity of line features

A system's false positive rate depends on the uncertainty regions' *selectivity* [12], i.e. the probability that it contains a spurious feature at random. For points, the selectivity is the uncertainty region's area divided by the area of the image. For lines, we need the fraction of positions and orientations of a spurious segment in the image that put the segment in the region.

Non-overlapping uncertainty circles

There is a set of translations that places a line segment of known length and orientation within the image, also given by its *configuration space* [18]. The line selectivity is the fraction of these translations that place the segment within the line uncertainty region, and can be obtained by shrinking both the region and the image along the segment's orientation and by its length. For shrinking the region along the segment's orientation, the shaded regions in Fig. 2 show two cases, distinguished by the image segment's orientation relative to that of the common outer tangent, θ_1 . An image segment's orientation is bounded by the orientation of the crossed common tangent, θ_2 . From Fig. 3,

$$\theta_1 = \sin^{-1} \frac{R-r}{L}, \quad \theta_2 = \sin^{-1} \frac{R+r}{L},$$

provided $L \geq R-r$ for θ_1 and $L \geq R+r$ for θ_2 . If the circles don't overlap, then $L \geq R+r$.

The set of translations is further constrained by the segment's length, obtained by shrinking the shaded region in Fig. 2 by this length. To compute the selectivity, we need the area of the shrunken region. While this area can be computed exactly [2], we use a more convenient rectangular box as an overestimate (Fig. 4). For comparison, Fig. 2 shows the box surrounding each corresponding line uncertainty region. From Fig. 4, after shrinking the rectangle along the base by ℓ , the region's area is

$$\text{If } \theta \in [0, \theta_1] \text{ \& } \ell \leq R+r+L \cos \theta, \\ A = (R+r+L \cos \theta - \ell)2r,$$

$$\text{If } \theta \in [\theta_1, \theta_2] \text{ \& } \ell \leq R+r+L \cos \theta, \\ A = (R+r+L \cos \theta - \ell)(R+r - L \sin \theta),$$

Note that $R+r-L \sin \theta \geq 0$, since $\theta \leq \theta_2 = \sin^{-1} \frac{R+r}{L}$.

For the image, the area of the region of translations for the same image segment is (Fig. 5)

$$A_I = (w - \ell \cos \theta)(h - \ell \sin \theta)$$

The selectivity of a random line segment of known length and orientation is $\frac{A}{A_I}$.

In general, there will be several line segments with different lengths and orientations that fall within a line uncertainty region. To account for orientation, we assume that random line segments are equally likely to fall at any angle. We then integrate A and A_I over their respective ranges of allowable orientations to get volumes of allowable positions of a random line segment (with known length). Integrating the two area formulas over an arbitrary range $[\omega_1, \omega_2]$ gives:

$$\begin{aligned} v_1(\omega_1, \omega_2) &= (R+r-\ell)2r(\omega_2 - \omega_1) \\ &\quad + 2rL(\sin \omega_2 - \sin \omega_1) \\ v_2(\omega_1, \omega_2) &= (R+r-\ell)(R+r)(\omega_2 - \omega_1) \\ &\quad + (R+r-\ell)L(\cos \omega_2 - \cos \omega_1) \\ &\quad + (R+r)L(\sin \omega_2 - \sin \omega_1) \\ &\quad - \frac{1}{2}L^2(\sin^2 \omega_2 - \sin^2 \omega_1) \end{aligned}$$

From the area formulas, the range of θ is divided into two intervals at $\theta = \theta_1$, and the length of the image segment constrains the range of orientations such that $\cos \theta \geq \frac{\ell - (R+r)}{L}$, or

$$\theta \leq \phi = \cos^{-1} \left(\frac{\ell - (R+r)}{L} \right).$$

Note that ϕ exists iff $R+r-L \leq \ell \leq R+r+L$. The first inequality holds if the circles do not overlap, and the second must be true for the image segment to fit in the uncertainty region (Fig. 1). From this, the volume V that corresponds to the two area formulas is given by

$$V = 2 \begin{cases} v_1(0, \phi) & \text{if } \phi \leq \theta_1, \\ v_1(0, \theta_1) + v_2(\theta_1, \phi) & \text{if } \theta_1 \leq \phi \leq \theta_2, \\ v_1(0, \theta_2) + v_2(\theta_1, \theta_2) & \text{if } \theta_2 \leq \phi, \end{cases}$$

where $\ell \leq R+r+L$. Integrating A_I from $\theta = -\pi/2$ to $\theta = \pi/2$ gives

$$V_I = \pi w h - 2\ell(w + h) + \ell^2.$$

The selectivity equals $\frac{V}{V_I}$.

In summary, a model line segment's selectivity can be computed as follows. Let r and R , $r \leq$

R , be the radii of the uncertainty circles for the endpoints of the line segment, computed using the technique of Section 3 or, for planar models, using the known analytic solution. Let L be the distance between the centers of the two circles, and let ℓ be the expected length of the corresponding image line segment. Then compute the selectivity μ from the above formulas for V , V_I , v_1 , v_2 , θ_1 , θ_2 , and ϕ .

This method assumes that the image line segment's length is known. Because in real images all lengths are not equally likely, we cannot integrate out the length dependence. Instead, if we can estimate the percentage α of the model that is occluded, then we can estimate occlusion by assuming that each model segment is occluded by α , giving $\ell = (1 - \alpha)L$ [10].

Overlapping uncertainty circles

Occasionally, the uncertainty circles may overlap, either by intersection ($R - r \leq L \leq R + r$) or inclusion ($L \leq R - r$). For these situations, [2] derives similar volume formulas, which are a little more complex and handle the few additional cases that arise over the range of θ .

2.3 Likelihood of false positives

The false positive rate of an alignment system can be computed from the expected selectivity of a feature ([10, 11, 12], also Section 4). To compute the false positive rate, let $\bar{\mu}$ be the expected selectivity, let s be the number of unmatched image features, let m be the number of unmatched model features, and let m' be the number of model point features that are used for generating hypotheses. If the s image features occur independently and at random, the probability of at least one image feature appearing in a propagated region with selectivity $\bar{\mu}$ is

$$p = 1 - (1 - \bar{\mu})^s.$$

The probability of at least k of the m propagated regions having at least one random feature is

$$w_k = 1 - \sum_{i=0}^{k-1} \binom{m}{i} p^i (1-p)^{m-i}. \quad (1)$$

w_k is the probability of a false positive of size k . If we match a fixed image triple to all possible

model triples, the probability that at least one match leads to a false positive of size k is

$$e_k = 1 - (1 - w_k)^{\binom{n'}{3}}. \quad (2)$$

3 A Study of Uncertainty Regions for 3D Point Features

To use these false positive formulas, we need the expected selectivity of point and line features. These can be computed from the propagated uncertainty regions for points, which we assume are circular. This section experimentally justifies this assumption, and proposes a method for computing the bounding circles. Specifically, Section 3.1 shows that the uncertainty regions generally can be approximated accurately with "uncertainty circles" centered at the nominal points, although at times the shapes of the uncertainty regions can be complex (Section 3.2). Section 3.3 demonstrates that the uncertainty circles can be computed precisely with a small amount of numerical sampling, so that the simple approach of sampling is both accurate and efficient.

3.1 Comparing the shapes of uncertainty regions to circles

To see how well uncertainty circles bound the errors in the image locations of predicted model points, this section runs two experiments that compare the true regions to the circular fits. The radii of the circles are computed using the maximum distance from the nominal point to a boundary point. To compare regions, we use the following error measure. Let A_t equal the area of the true region, and let A_c equal the area of the approximating circle. The error measure is

$$\frac{A_c - A_t}{A_c}, \quad (3)$$

where the sign is used to discriminate which area is larger. Since it is based on the difference in areas, the measure will be large when the fit is poor. Since the difference in areas may be large if the perimeters do not line up exactly, the measure may also be large when the fit is relatively good. Thus, the measure provides a conservative estimate of the badness of the circular fit.

Experiment 1: Accuracy of uncertainty circles for random models

This experiment examines how often we can expect the uncertainty circles to be correct, in particular, how often the maximum error is 1%, or 10%. Also, we estimate what the maximum error will be 90%, 95%, and 99% of the time. To do this, we run a series of trials of an alignment algorithm and compute the error measure (Eq. 3) for each. The percent of time the error satisfies some criteria is estimated by the fraction of trials over which the error measure satisfies that criteria. For the algorithm, we assume that the image points effectively arise at random, which is reasonable if the image has significant clutter.

Method: We ran 100 trials where a model is projected into an image and the error measure of Eq. 3 is computed for each model point. In each trial, a random triple of image points is matched to a random triple of model points taken from a randomly-generated model (for details see [2]). The three-point match is used to project the model into the image, which gives the nominal image locations of the model points. Except for model points in the plane of the matched model points, there are two possibilities for each nominal image location [15, 1].

Using $\epsilon = 5$, the ϵ -circles around the three image points are sampled uniformly at 25 points each. Every triple of sampled points is matched to the three model points, and used to compute the image locations of all the model points. This results in a region in the image for each model point. The area of each region is computed by counting the pixels within the region's boundary (see [2]). The radius of the corresponding uncertainty circle is obtained by taking the maximum distance from the nominal point to a boundary point.

As noted, there are two solutions for each pair of model and image triples, which correspond to a reflection about any plane parallel to the image [15, 1]. From [1], let H_1 and H_2 be the differences in the z coordinates between the first model point and the second and third model points, respectively; $-H_1$ and $-H_2$ for the reflected solution. To distinguish the two solutions, we use the values of H_1 and H_2 that occur when the matched image points are at their nominal locations. If the nominal H_1 is larger, we take all solutions with the same sign for H_1 as being from the same region. We do the opposite if the nominal H_2 is

larger. This method separates the two regions corresponding to the two weak-perspective solutions, unless they overlap. If they do overlap, there really is one region, and this method splits it.

Results and Discussion: Over the 100 trials, 1163 uncertainty regions were tested. The average area was 583.53 for the correct uncertainty regions and 662.43 for the approximating circles. For 96.73% of the uncertainty regions, the error (using the error measure) between the true region and the approximation was less than 1%, and for 97.94% of the uncertainty regions the error was less than 10%. Also, the maximum error for 90% of the regions was 1%, for 95% of the regions was 1%, for 98% of the regions was 10%, and for 99% of the time 51%. This suggests that uncertainty circles are generally very accurate.

Experiment 2: Accuracy of the uncertainty circles for the telephone model

For comparison, we ran the same set of trials on a model of a telephone (Fig. 6).

Method: The method is as in Experiment 1, but using the telephone model at every trial.

Results and Discussion: For 100 trials with the phone model, 1092 uncertainty regions were generated. The average area was 495.59 for the correct uncertainty regions and 450.13 for the approximating circles. Notice that this time the average area for the overestimates is lower than for the exact areas. This is because the method used to compute the true regions can overestimate them a little when the fit is good, an effect which turned out to be stronger than the overestimate in the circular fit, because very few of the circular fits were poor. Here, for 98.01% of the uncertainty regions the error between the true region and the approximation was less than 1%, and for 99.08% of the regions it was less than 10%. The maximum error for 90%, 95%, and 98% of the regions was 1%. Further, for 99% of the regions the maximum error was 10% instead of 51%. So it appears the circular fits work better for the specific model of a telephone.

3.2 Cases where errors are greatest

Of the 100 trials of random models, two had errors greater than 25%. Fig. 7 displays, relative to the image, the uncertainty regions and uncertainty circles corresponding to the largest errors for those

trials (78.8% and 81.7%). In both cases, the corresponding angles between the matched model and image points were very close. Geometrically, this means the plane of the model points was almost parallel to the image, a situation which is inherently unstable [1]. The unusual shapes of the true uncertainty regions in Fig. 7 are due to the computation of the uncertainty regions (Section 3.1), and represent cases where the two regions overlap and are split in two.

For the phone model, only one trial had errors greater than 25%, specifically 27.0%. The uncertainty region and uncertainty circle are shown in Fig. 7 (same scale as other examples).

From the cases with large errors, we can infer that, in an alignment system that tries many or all pairs of point triples for aligning a model to the image, situations with large errors could be avoided by checking whether the angles between the points are similar. However, this may lead to relying on an arbitrary threshold. As a consequence, it would be better to handle these cases specially by sampling extensively and then walking the boundaries of the resulting regions.

3.3 Computing uncertainty circles efficiently

Given that circles centered at the nominal points approximate well the uncertainty region boundaries, all that is needed is to compute the radii of the circles. A simple approach is to sample points from the error circles around the matched image points and take the maximum distance from the predicted nominal point as the radius. This process will be efficient if few sample points are required. This section infers how few points are needed.

Experiment 3: Using fewer sample points for random models

To see how few sample points are needed, this experiment tests, for various numbers of points, n , and for a series of trials, the percent of time (fraction of trials) that the error in using n points is less than some limit. This is compared to using 25 points, as in the last two experiments.

Method: A series of 100 trials were ran using random image triples matched to random model triples from randomly-generated models, as in Experiment 1. For each trial, the error circles around

the matched image points are sampled uniformly at 25 points and 10 points. For each propagated uncertainty region, the error in using the smaller number of samples to using 25 samples is computed. This is repeated for 9, 8, and 7 sample points as well.

Results and Discussion: The results are shown in Table 1. Note that the percentages do not strictly decrease as fewer sample points are used. This is because the circles around the image points are sampled uniformly, so that using different numbers of sampled points can give different samples on the circles. Hence, when the percentages are close, there may be cases where fewer sample points do better. Nevertheless, this effect should be small. Notice that the average percent error does indeed increase monotonically.

We can use Table 1 to pick a reasonable number of points for sampling the image error circles. If we permit 5% error, then using 8 sample points instead of 25 should be accurate over 99% of the time. Also, the average error in using eight points is very small (1.137%).

A better feel for how accurate is the use of fewer sample points is given by statistics on the radii, shown in Table 2. From the table, the average difference in the radii for eight sample points was .08 pixels, and the worst case difference was 3.24 pixels. Relative to the radius for twenty-five points, the average difference is .575%, and the maximum difference is 8.96%.

Experiment 4: Using fewer sample points for telephone model

Method: This experiment is the same as Experiment 3, except that the phone model is used.

Results and Discussion: Tables 1 and 2 give the results. From Table 1, we again can use eight points to limit errors to 5% over 99% of the time. From both tables, it appears that using fewer sample points works slightly better with the phone model than with random models.

To illustrate the use of uncertainty circles, Fig. 8 shows an example of the propagated uncertainty circles, where eight sample points were used. The three smallest circles correspond to the assumed errors in the matched image points, which in this example were matched correctly. For the unmatched model points, the other circles show the regions to be searched for matching image points.

The self-occluded model points were removed beforehand. Still, some of the remaining corner points are occluded by other objects, and the uncertainty regions provide a means to reason that this is so after a relatively small amount of search in the image.

Notice that the sizes of the propagated uncertainty regions vary considerably for different model points. Consequently, an approach that relies on fixed-sized error bounds, as in [15], can lead to correct matches being missed (when the bounds are too small), and incorrect matches being accepted (when the bounds are too large and include spurious image points).

4 Measuring the Sensitivity to False Positives

4.1 Expected selectivity of point features

We now use the analyses of Sections 2 and 3 to examine alignment's sensitivity to false positives. For point features, the expected selectivity has been used before to analyze false positive rates for alignment where the models are flat [12], and also for alignment with solid models but using a different uncertainty propagation technique [11]. We can use the expected selectivity to compare the uncertainty propagation technique used here to the one in [11].

For flat models, the propagated uncertainty regions can be computed exactly. It would be interesting to see how much the chance of a false positive increases from planar to solid models, since the propagated uncertainty regions are larger for points out of the plane of the matched model points than for their corresponding points in the plane [1]—for a 3D point, the corresponding point in the plane is the intersection of the plane and the perpendicular from the plane to the 3D point. Again, we can use the expected selectivity for the comparison.

Experiments 5 and 6: Expected selectivity of point features

Method: To compute the expected selectivity, we re-ran 1000 trials of the same type as in Experiments 3 and 4, except five was added to each radius before computing the area, in order to account for expanding each uncertainty region outwards by $\epsilon = 5$ pixels.

Results and Discussion: Using random models with 8 sample points over 1000 trials gave 11349 propagated regions with average area 973.25 square pixels. Using the phone model with 8 sample points over 1000 trials gave 11085 propagated regions with average area 979.78 square pixels. The resulting selectivities along with those for [11] and [12] are shown in Table 3.

The expected selectivity for the uncertainty circles is about half that for [11], which implies that the uncertainty circles should give significantly better performance. Furthermore, it appears that the selectivities of solid models are only slightly greater than for planar ones. We can infer from this that, when point features are used, recognizing solid objects with alignment is a only little more sensitive to false positives than recognizing planar objects.

4.2 Expected selectivity of line features

Experiment 7: Expected selectivity of line features for the telephone model

Method: To compute the expected selectivity, we used the formula given in Section 2.2. We ran a series of the same trials from Experiments 5 and 6 when the selectivity of point features was computed. For each trial, we used each pair of uncertainty circles that corresponds to a line segment in the telephone model (Fig. 6) and computed the line segment selectivity. This was repeated for various amounts of occlusion, α .

Results and Discussion: For 1000 trials, the selectivity of 9560 line uncertainty regions was computed and averaged. Table 4 gives the selectivities for different amounts of occlusion. As expected, the selectivities for lines are much less than for points (compare to Table 3).

4.3 Limits on Scene Clutter

A recognition scheme based on extended model features will fail if a scene becomes extremely cluttered. It would be useful, then, to know how much clutter a recognition system can accommodate before the probability that it will fail is significant. We can use Eq. 2 to estimate this limit. Specifically, given an image triple, we can compute the maximum value of s such that $e_k \leq \delta$, where δ is a preset limit, and $k = fm$ for some fraction f of the model features.

Table 5 shows the results for $\delta = .001$ (the .01 and .0001 cases are similar). The limits for the uncertainty propagation technique of [11] are very low. Although the numbers are greatly improved using uncertainty circles, it is only when line segments are used that numbers of features are in the range of images with substantial scene clutter (≈ 500 features).

4.4 Threshold for Accepting a Partial Match

When the extended features of a model are used for verification, we want to know how many must be matched before we can stop looking for more matches. We can use Eq. 1 to set a threshold such that the chance that a false positive will arise is less than a preset limit. Let f be the percentage of model features that must be matched to keep the probability of a false positive at most δ_2 . Substituting mf for k , we want to find the minimum f such that $w_{mf} \leq \delta_2$. Table 6 shows the results for line segments. As a check on the method, the recognition system of [15] used $f = .5$ as a threshold on the percentage of the model to verify. In the examples given, anywhere from 0 to 50% of the model was occluded, and so the thresholds predicted by the table are in approximate agreement with the experimental threshold chosen.

5 Conclusion

An important criteria for a recognition system is that we can trust it when it decides if an instance of the model occurs in the image. Once the system is guaranteed to not mistakenly discard any correct hypotheses, its usefulness is determined by its sensitivity to false positives. This paper gave a theory for analyzing the false-positive sensitivity of alignment-style recognition systems (Section 2). The main contribution is a formula for the selectivity of line features (Section 2.2). A feature's selectivity can be used to infer the expected performance of recognition systems. It can also be used to set a threshold on how much of a model must be identified in an image before the object is recognized (Section 4.4).

We also provided an error analysis of point features for alignment-style recognition of 3D models from 2D images (Section 3). The earlier analysis of [11] was conservative in its bounds on the propagated uncertainty, and Section 3 showed we can

do better. In fact, the analysis is almost always a solution, which means its bounds are exact, except where the 3D pose solution is inherently unstable. In these cases, the bounds conservatively overestimate the exact bounds.

Even though the error propagation technique in Section 3 is generally accurate, the technique has the disadvantage of being numerical. For most recognition problems, however, the time to compute the solution is effectively constant, as though the solution were analytic.

These contributions tie together well for building a fast and robust alignment system. The uncertainty analysis provides the correct minimal search regions to guarantee that no correct hypotheses are lost. Further, the uncertainty regions can be computed quickly using the error propagation technique and a fast solution for the image position of an unmatched model point. Once computed, the uncertainty regions usually are small enough to be searched rapidly for candidate image features. Then the current hypothesis can be evaluated, using a predetermined threshold on the percentage of model features that must be matched.

References

- [1] Alter, T. D., "3D Pose from 3 Corresponding Points Under Weak-Perspective Projection," MIT A.I. Lab. Memo 1378, July 1992.
- [2] Alter, T. D., "Fast and Robust 3D Recognition by Alignment," Master's Thesis, MIT Dept. Elec. Eng. and Computer Sci., September 1992.
- [3] Ayache, N., & O. D. Faugeras, "HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, no. 1, pp. 44-54, January 1986.
- [4] Ballard, D. H., "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111-122, 1981.
- [5] Basri, R., & S. Ullman, "The Alignment of Objects with Smooth Surfaces," in *Proc. Second Inter. Conf. Computer Vision*, pp. 482-488, 1988.
- [6] Bolles, R. C., & R. A. Cain, "Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method," *Inter. J. Rob. Res.*, vol. 1, no. 3, pp. 57-82, 1982.
- [7] Cass, T. A., "Feature Matching for Object Localization in the Presence of Uncertainty," in *Proc. Third Inter. Conf. Computer Vision*, pp. 360-364, 1990.

- [8] Fischler, M. A., & R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. ACM*, vol. 24, pp. 381-396, June 1981.
- [9] Grimson, W. E. L., & D. P. Huttenlocher, "On the Sensitivity of the Hough Transform for Object Recognition," *IEEE Trans. Pat. Anal. Machine Intell.*, vol. 12, no. 3, March 1990.
- [10] Grimson, W. E. L., & D. P. Huttenlocher, "On the Verification of Hypothesized Matches in Model-Based Recognition," *IEEE Trans. Pat. Anal. Machine Intell.*, vol. 13, no. 12, December 1991.
- [11] Grimson, W. E. L., D. P. Huttenlocher, & T. D. Alter, "Recognizing 3D Objects from 2D Images: An Error Analysis," in *Proc. IEEE Conf. Computer Vision Pat. Rec.*, pp. 316-321, 1992a.
- [12] Grimson, W. E. L., D. P. Huttenlocher, & D. W. Jacobs, "A Study of Affine Matching with Bounded Sensor Error," in *Proc. Second European Conf. Computer Vision*, pp. 291-306, 1992b.
- [13] Grimson, W. E. L., & T. Lozano-Pérez, "Model-Based Recognition and Localization from Sparse Range or Tactile Data," *Int. J. Robotics Res.*, vol. 3, no. 3, pp. 3-35, 1984.
- [14] Horaud, R., "New Methods for Matching 3-D Objects with Single Perspective Views," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, no. 3, pp. 401-412, May 1987.
- [15] Huttenlocher, D. P., & S. Ullman, "Recognizing Solid Objects by Alignment with an Image," *Inter. J. Computer Vision*, vol. 5, no. 2, pp. 195-212, 1990.
- [16] Jacobs, D. W., "Optimal Matching of Planar Models in 3D Scenes," in *Proc. IEEE Conf. CVPR*, 1991.
- [17] Lainnainmaa, S., D. Harwood, & L. S. Davis, "Pose Determination of a Three-Dimensional Object Using Triangle Pairs," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, no. 5, pp. 634-647, Sept. 1988.
- [18] Lozano-Pérez, T., "A Simple Motion-Planning Algorithm for General Robot Manipulators," *IEEE J. Rob. Aut.*, vol. 3, no. 3, June 1987.

	1%	2%	3%	4%	5%	6%	ave	max	min
10	72.06	93.12	98.61	99.13	99.22	99.56	0.684	21.97	-.34
9	57.27	84.16	98.00	98.70	98.87	99.04	1.031	31.60	-.37
8	55.61	75.98	90.43	98.87	99.39	99.57	1.137	17.12	-.49
7	46.30	63.27	77.89	91.65	97.91	98.61	1.670	25.53	-.31
10	66.40	91.91	99.37	99.55	99.64	99.64	0.726	8.55	-.33
9	60.83	84.19	98.02	99.46	99.55	99.55	0.913	13.19	-.33
8	62.15	91.91	99.37	99.55	99.64	99.64	0.981	11.76	-.30
7	46.46	64.24	80.32	92.81	98.38	99.55	1.532	12.80	-.33

Table 1: Percentage of time error was less than 1%-6% for different numbers of sample points, plus average, maximum, and minimum percent errors over all trials. Top: using 1149 propagated uncertainty regions from random models. Bottom: using 1113 uncertainty regions from the telephone model.

	ave	max	min	ave percent	max percent	min percent
10	.05	2.55	-.05	.344	11.67	-.17
9	.08	3.87	-.03	.521	17.30	-.18
8	.08	3.24	-.05	.573	8.96	-.24
7	.13	4.21	-.02	.844	13.70	-.16
10	.05	0.69	-.05	.365	4.37	-.17
9	.06	1.30	-.02	.459	6.83	-.17
8	.07	1.12	-.03	.494	6.07	-.25
7	.10	1.23	-.03	.772	6.62	-.16

Table 2: Differences in radii for different numbers of sample points. Top results are for random models, bottom results are for the telephone.

Method	Models	Model type	Uncertainty	$\bar{\mu}$
Uncertainty Circles	Random	Solid	Circular	.003722
Uncertainty Circles	Phone	Solid	Circular	.003747
Grimson et al. 92a	Random	Solid	Polygonal	.00866
Grimson et al. 92b	Random	Planar	Circular	.002911

Table 3: Expected selectivities of point features.

α	0.00	0.25	0.50	0.75	1.00
μ	.000647	.001017	.001311	.001550	.001750

Table 4: Expected selectivities of line features for different amounts of occlusion, α , using the telephone.

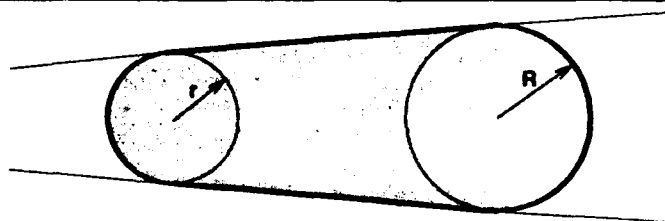


Figure 1: Region to search for candidate line segments

Method	α	$f = 0.25$	0.50	0.75
Line Uncertainty Regions	0.00	161	537	1200
Line Uncertainty Regions	0.25	102	341	763
Line Uncertainty Regions	0.50	79	265	592
Line Uncertainty Regions	0.75	67	224	500
Uncertainty Circles	-	31	97	216
Grimson et al. 92a	-	15	43	95

Table 5: Approximate limits on the number of sensory features for different amounts of occlusion α and different fractions f of model features used. Table is for $\epsilon = 5$, $\delta = .001$, for lines $m = m' = 200$ (line uncertainty regions), and for points $m = 197$ and $m' = 200$ (uncertainty circles and [11]).

δ_2	$\alpha = 0.00$	0.25	0.50	0.75	1.00
0.01	.36	.49	.57	.63	.67
0.001	.38	.51	.60	.66	.70
0.0001	.41	.54	.62	.68	.72

Table 6: Predicted termination thresholds for different amounts of occlusion α , and for different limits δ_2 on the false positive probability. Table is for $\epsilon = 5$, $m = m' = 200$, and $s = 500$.

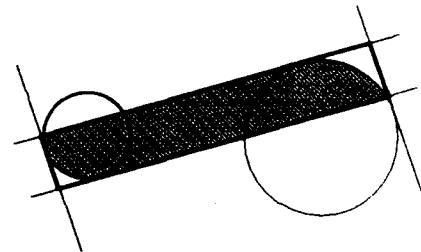
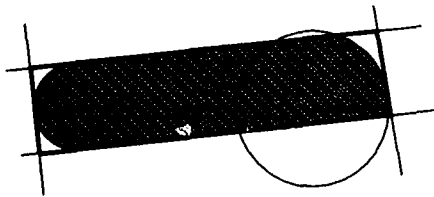


Figure 2: Region of translations with orientation constraint and rectangular bound.

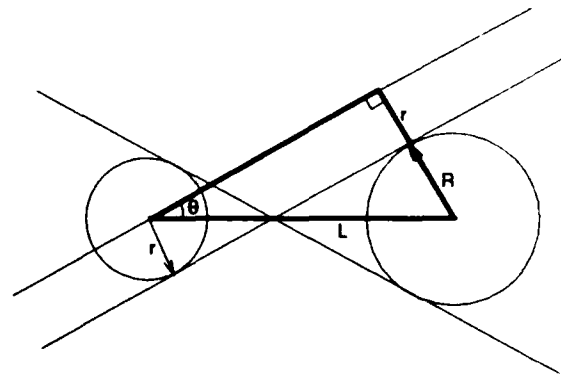
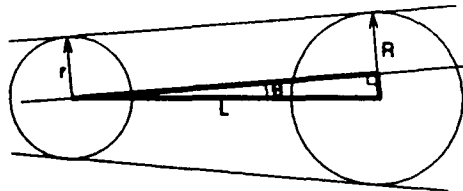
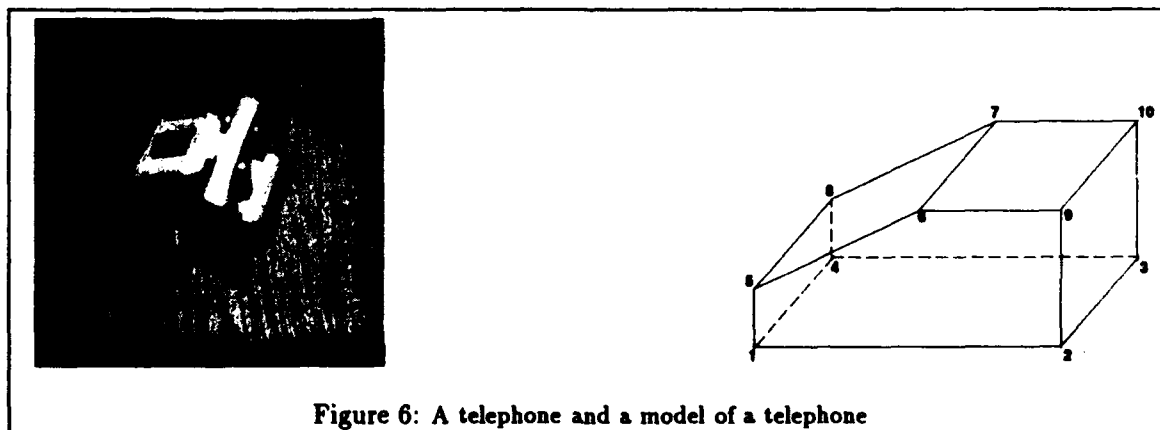
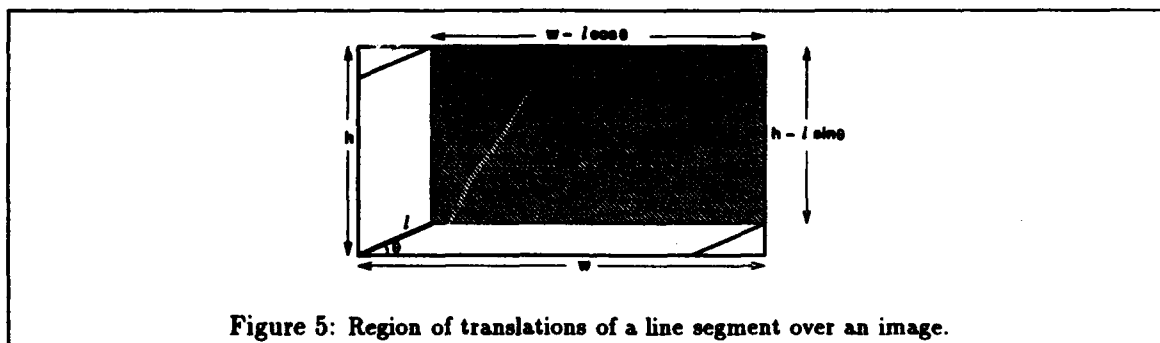
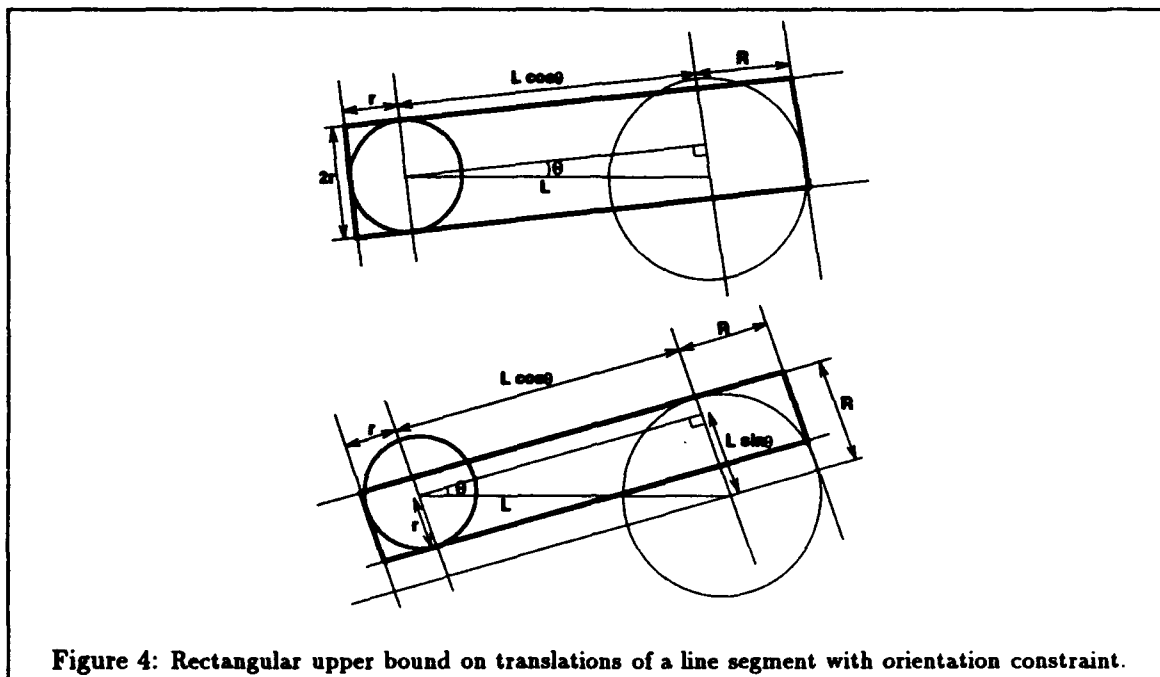
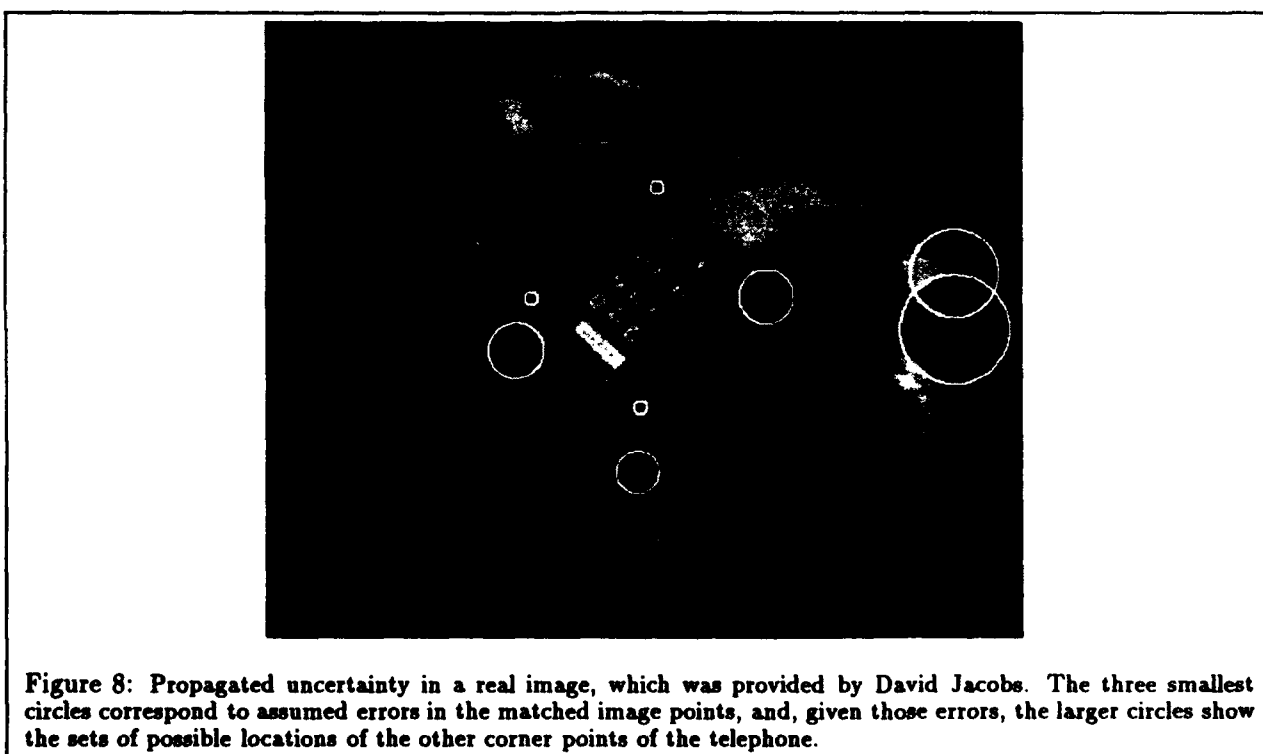
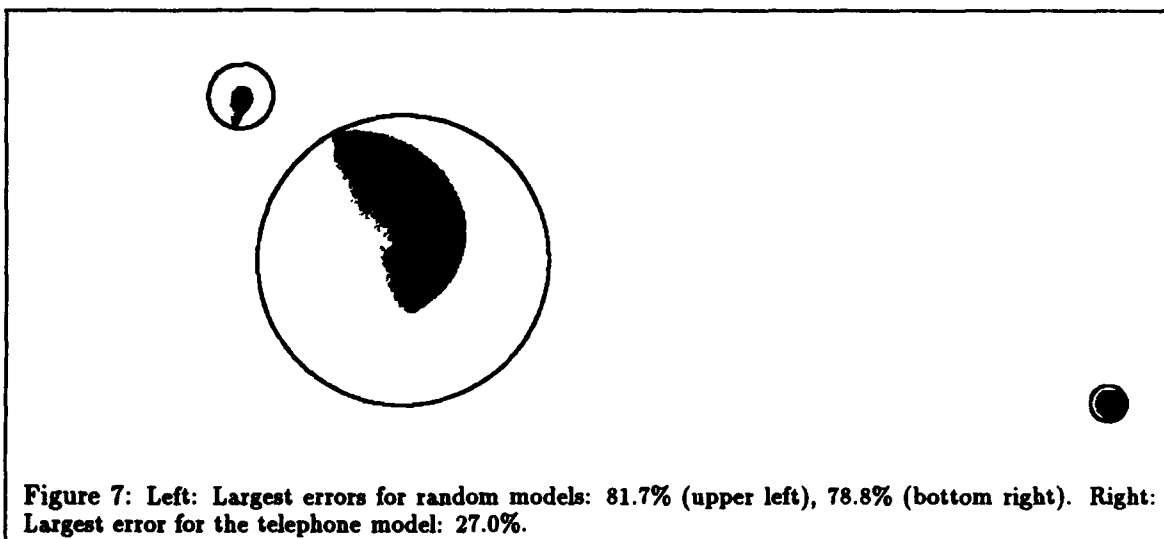


Figure 3: Orientations of the common outer tangents (left) and the common cross tangents to the circles (right), which give the maximum possible angle of a line segment with an endpoint in each circle.





Alignment Using An Uncalibrated Camera System

Billibon H. Yoshimi
Peter K. Allen

Center for Research in Intelligent Systems
Department of Computer Science
Columbia University
New York, NY 10027

Abstract

Calibration of cameras and manipulators for robotic tasks is a difficult and sensitive process. We present a technique that uses active camera motion to recover image space properties that can be used to accurately control and position a robot hand/eye system that uses an uncalibrated camera. The algorithm is verified by an experiment where a robot completes the task of inserting a peg into a hole with an error of 3mm.

This work was supported in part by DARPA contract DACA-76-92-C-007, NSF grants IRI-86-57151, CDA-90-24735, North American Philips Laboratories, Siemens Corporation and Rockwell International.

1 INTRODUCTION

In many real world applications, there is a need to perform alignment tasks between two objects. Two simple, generic tasks are inserting a peg into a hole and aligning objects into arbitrary geometric configurations (e.g. robotic assembly tasks.) A key component of this problem is positioning where there is little room for mechanical error. The idea of precision measurement (in our example, alignment) using a mechanical device, photographic emulsions or photo-electric sensors, has been examined in great detail by the researchers in non-topographic photogrammetry. By using models which account for most of the

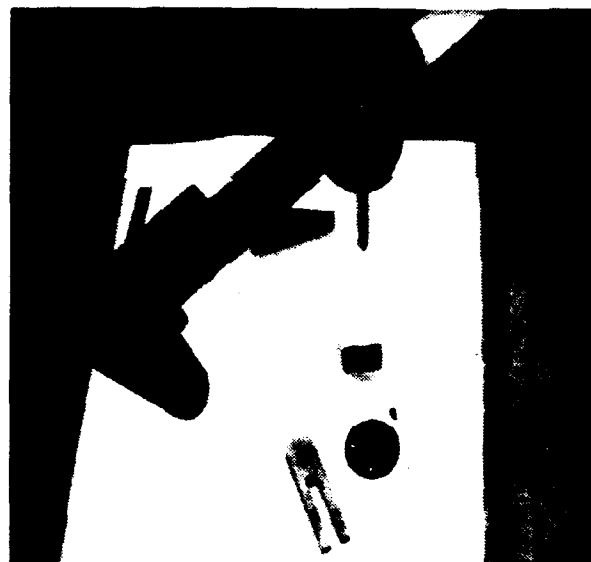


Figure 1: View of camera, robot, and multiple target setup

aberration and lens defects in modern lenses, they obtain highly precise calibrations of their camera systems. For more information see Karara[6]. These methods are often difficult to understand and inconvenient to use in most robotics environments. They usually require the minimization of several, complex, non-linear equations of multiple variables (of which the results are not guaranteed to be robust.) Other methods for performing camera calibration for robots include the works of Tsai [16, 15], Young et. al. [18], Bennett et. al. [1], and Holt et. al.



Figure 2: View of targets from camera

[4] for example.

To give the reader an idea of the alignment/insertion task, figure 1 shows our experimental setup. Off the end of the end effector of our robot is a probe with a sharp tip (the "peg".) The target in this scene is a 2mm hole in the machined aluminum block located almost directly below the probe. Figure 2 shows a view of the target objects taken from the camera system. In this figure, the holes in the machined block are more easily seen. The goal of the task is to maneuver the probe to a position where it is directly above the target, and then to insert the probe into the target.

Another class of methods revolves around the depth from motion paradigm. This body of research tries to recover the absolute pixel velocity for objects in image space. Here too the researchers are searching for an absolute transformation from a known reference (the velocity of a known object) and an unknown system (the actual, time-varying, intensity data). The method we propose does not require the absolute positional information that both of the aforementioned systems require. It uses simple image displacement data (generated from the movement of the camera system) to gen-

erate an estimated position where it expects that the object motion will be minimized with respect to the camera movement.

Our technique takes the typical mapping from 3-D positions to image coordinates, and instead of finding this mapping, it recovers a property of the image coordinates. The traditional mapping problem (known as the calibration problem) determines the position of objects based on relative scale difference, perspective distortion, and/or several other properties which exist between a calibrated system and an observed system. These positional values can be obtained from both static and dynamic systems. These methods do not exploit the fact that a known movement in the camera system can result in useful motion information in the image system without knowing the exact calibration between the systems.

We approached the problem by asking the following question: How can I get a robot to perform a given task using only uncalibrated visual input to direct the robot's actions? In many cases, it is not necessary for the robot to have a completely calibrated work area. (It is not necessary to know the exact positions of everything in the robotic workspace. It may be more important to know only the exact position of certain items.) We propose a new technique, similar to the work of Sawhney [11, 12], which will allow the robot system to maintain an arbitrary, geometric relationship with an object system, and as a result of certain operations, the robot-object system can "calibrate" itself to or "can define its location with respect to" the unknown camera system. The newness of our technique arises from the fact that our system performs the useful task of moving to the goal position without ever really knowing the true location of the camera system.

2 OVERVIEW OF METHOD

In order to perform the peg-in-hole insertion task, we broke the task into two parts: the alignment task and the actual insertion task. The alignment task serves the end effector

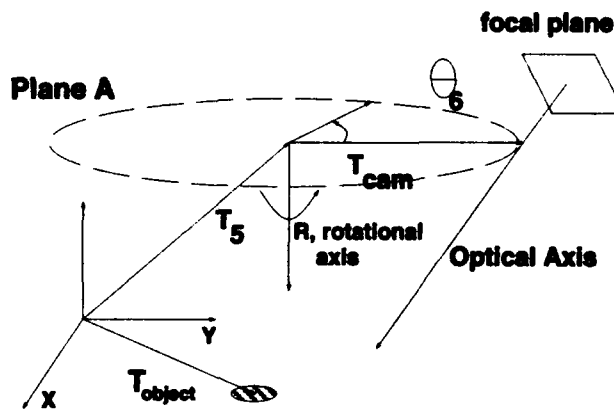
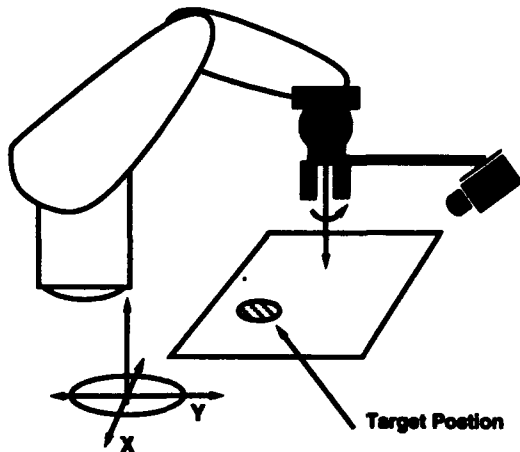


Figure 3: Experimental setup

in a plane in robot space until the alignment condition occurs (that being when the object to be servoed to and the end effector lie on the same axis.) The insertion task relies on the fact that the alignment stage has constrained the solution to lie along a line (thus making the insertion task simply a one degree of freedom search.)

A simplified setup is shown in figure 3. The task is to maneuver the end effector to a position directly over the target position.

We started our investigation by examining what would happen if we attached some sensing system to the rotational axis, such that the system could image the rotational axis. We noticed the following effect as we servoed the rotational joint over a small angle (see figure 4.) Those objects which were

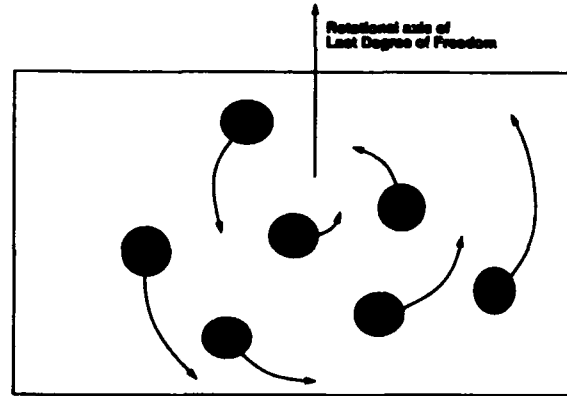


Figure 4: Demonstration of the observed effect

further away from the axis of rotation moved a greater distance than those points closer to the axis. This effect is not new and is very similar to the work done by researchers on the analysis of the Focus of Expansion for time to impact studies.

We make the simplifying assumption that the objects do not change their appearance as we perform the rotation. One simple way of doing this was to use point-like targets. The point-like targets rely on the fact that the perspective distortion is highly localized due to the fact that the targets have a high level of spatial coherence.

Using the effect noticed above, we transformed the alignment problem into one of a positioning problem in a plane. The simplification is justified by the following observations:

1. The initial movements of the manipulator-camera system are in the $X - Y$ plane. The projection of the Z -distance to the object on the rotational axis is kept constant.
2. Once the alignment has been performed in $X - Y$ space, the only movement necessary is a pure translation along the rotational axis (for our scenario, the Z -axis).

To perform our peg-in-hole insertions we also make the following assumptions:

- T_5 (the transform from the world coordinate system to the end-effector of the

robot, minus the last rotational degree of freedom) is known.

- R (the last degree of rotational freedom) is known.
- T_{cam} , the camera transform matrix, is unknown.
- P , the perspective effect introduced by the camera system where the missing parameter is λ , the focal length, is unknown
- Once the camera was positioned, it remained fixed with respect to the rotational axis of the 6th joint of the PUMA. The camera was not allowed to move in its mount nor was the mount allowed to move with respect to its mounting point on the robot.
- The camera can image the target object during any servoing operation. There should not be a time where the object leaves the focal plane.
- and finally, T_{object} , is unknown.

The following constraints were not necessary:

- intersection of the optical and rotational axes. Proof: If you imagine the system setup (camera, extending rod, gripper and task system) where each piece is immovable, you will notice that the rotational axis projects to a line in the camera imaging area. This line, by definition of the rigid system, cannot change. Its position is dictated by a fixed projection and since the line simply rotates around its own symmetrical axis, its position does not change. Note: if the rotational axis is seen to have translated, it means that the manipulator did not go through a simple rotation, but may actually have translated.
- knowledge of the focal length, or
- knowledge of T_{cam} .

In the figure 3, the robot-camera system is constrained to move in the plane A , where A is defined by the circle swept by the camera around the rotational axis, R . We have simplified the alignment task to one of

a 2 DOF problem. The goal state is one where the object simply rotates in the image plane without translating, hence satisfying the alignment condition which is that the object lies on the rotational axis. The rotational degree of freedom is used as a free variable for the alignment task and does not contribute to the final alignment state (for circularly symmetric objects).

Once an object has been selected in the camera's view, the robot rotates the camera around its rotational axis, R . By slowly rotating the camera around its rotational axis, we remove the correspondence problem (the object moves only a slight bit between consecutive shots, therefore making the correspondence between two shots trivial to compute.) If the only movement in the robot-camera system is caused by the rotation, the object will trace out a conic section, an ellipse under certain conditions¹, in the camera system. We propose to use these elliptical parameters to recover the alignment condition. One simple method requires that we move about in the plane A , sweeping out ellipses in camera space. The further away the object is from the rotational axis, the larger area is swept out by its ellipse projection into camera space. The closer we come to aligning the object to the rotational axis, the smaller the projected ellipses will become. The goal in this scenario is to devise a method for maneuvering the end effector's position in plane A to the position which causes the object to project to an ellipse with the smallest area.

¹ The degenerate conditions are if the object:

1. is directly orthogonal to the image plane - a circle,
2. is already aligned to the rotational axis - a point,
3. incident with a plane containing the focal point which is parallel to the image plane - parabola,
4. passes through a plane containing the focal point which is parallel to the image plane - hyperbola. and
5. lies in the same plane as the optical axis (where the optical axis and the rotational axis are perpendicular to one another) - line.

3 RECOVERY OF ELLIPTICAL PARAMETER DATA

The majority of this work was inspired by Safaee-Rad et. al. [10, 9, 14], Haralick [2], Magee et. al. [7], Sawhney et. al. [11, 12] and Shiu et. al. [13].

While inspired by these methods, we have developed a new formulation for deriving ellipses from scattered point data. In our current scenario, we accumulate the $(\theta_6, X \text{ projection}, Y \text{ projection})$ triplet derived from combining the angle made between the end effectors zero position and the current position of the end effector and the projection of the tracked feature into camera space. We then parameterized the curve traced out by the feature as:

$$x(\theta) = A \cos(\theta_6) + B \sin(\theta_6) + C \quad (1)$$

$$y(\theta) = D \cos(\theta_6) + E \sin(\theta_6) + F \quad (2)$$

The area enclosed by this curve (computed using Green's Theorem) is

$$(A^2 + B^2 + C^2 + D^2) \frac{\pi}{2}. \quad (3)$$

The full proof that the parametric curves generated by these equations are ellipses is contained in [17].

The problem of fitting raw data points to elliptical data was covered in both the Sawhney and Safaee-Rad works cited earlier. We were concerned primarily with developing a method which did not require data points be taken from the entire ellipse and which could be solved linearly. In our experiments, the elliptical data was taken over a 90 degree sector of the ellipse. Using only this data, we were able to fit ellipses quite well (see figure 5.)

4 THE SIMPLE SIMPLEX SEARCH METHOD

This method uses a version of the simplex method for finding local minima. We were motivated by the fact that the solution surface was fairly smooth and by the idea that even a simple "walking" algorithm should

be able to find the solution. We proposed creating a "walker" with three legs: a simplex (for two dimensional "walking") requiring three starting points. A simplex was created in X - Y space from the set of three arbitrary, non-collinear positions $((0.0, 0.0), (0.0, 50.0), \text{ and } (50.0, 50.0))$. The search method using the simplex simply "walks" down the surface by tossing the "leg" which is furthest uphill an equal amount downhill. Once it constrains the solution to lie between its "legs" it shrinks itself and tries "walking" down the surface using its new position and new, smaller "legs." This method is similar to the Simplex method of Nelder and Mead[8].

The implemented version of the algorithm for the simplex search runs as follows:

1. Initialize simplex. The robot moves to each position in the initial position set in turn. At each position, the robot tracks the movement of a point-feature in the image plane as the robot changes its value of θ_6 . After accumulating the object positions in the image plane (2-D point data), the computer fits a least-squares conic section to the points. From the conic section parameters, it computes the area of the elliptical trajectory taken by the projection of the object in the image plane. This process is performed once at each robot position given in the initial simplex set.
2. While none of the areas is less than a predetermined threshold (where zero area indicates a perfect alignment between the object and the rotational axis),
 - (a) Find the point, p_1 in plane A , whose ellipse encompasses the greatest area.
 - (b) Reflect the point, through the line connecting the other two points, p_2 and p_3 .
 - (c) Evaluate the area of the ellipse at the new point. If the new point's area is larger than the area of the point which generated it, you've trapped the minimum, so decrease the area of your search space.

The above algorithm tries to trap the global minimum using large simplex movements to surround the minima and when the minima is trapped, it reduces its search space by moving the point with the largest area to the middle of the simplex (roughly reducing the bounding area by one-third). The algorithm is repeated until the area of the ellipse computed for a position is falls below the area threshold.

5 IMPLEMENTATION

In figure 3, we show a schematic of the system set up for testing the new alignment method. We mounted a Sony XC-77 CCD camera in a bracket system off the end effector of a Puma 560 robot. The camera was not calibrated or position constrained when initially placed. The system was controlled using RCCL and RCI [3]. The images were digitized at 256x242 resolution and 8 bits gray scale at standard NTSC frame rates using the PIPE parallel image processing engine [5]. The resulting images were thresholded to recover a simple black object on a white background. In general, any recovery method can be used to generically extract object information from an image array. The object was positioned so the robot would not encounter singularities when moving to the new control positions.

Given that the only information necessary to constrain the alignment is the area of the projected ellipse on the image plane, it is not necessary to know anything about the geometry of the sensor setup.

6 RESULTS

In the experiment, we used the modified simplex method (see section 4) with an initial simplex of $((0.0, 0.0), (0.0, 50.0), \text{ and } (50.0, 50.0))$.

We built a feature tracker which assumes velocity constrained object motion in image space. At the beginning of the experiment, a scene was extracted by the image processor and the user was prompted to move a pointing device to the location of the

feature. The feature extractor used a Sobel operator with a fixed threshold to extract the predominant feature in the selected region. The tracker would follow the feature over consecutive image frames as long as the feature moved only small distances.

After establishing the feature tracker, the robot was instructed to move to the first position, stop, and rotate its last joint 90 degrees over the course of which it would extract 16 images spaced equi-angularly with respect to the robot's rotation.

The feature tracker tracked the movement of the selected target position over the complete 90 degrees (reporting to the controller the position of the object at 16 equi-angular positions over the duration of the movement.)

The centroid of feature (in image coordinates) was then fed to a least squares estimator to recover the ellipse parameters associated with the moving features's trajectory. These parameters were then fed into formula 3 for computing the area of the ellipse.

The process was repeated for the remaining two points in the simplex. We initialized the simple simplex algorithm using these three areas and allowed it to step its way to the minima.

The halting condition was when the area of the ellipse formed from a position was $\leq 1\text{pixels}^2$. The following table tabulates the results of this experiment:

#	X	Y	Area
I	0.000000	0.000000	1951.378696
I	-50.000000	0.000000	5811.561407
I	-50.000000	-50.000000	4679.246920
3	0.000000	-50.000000	284.406540
4	50.000000	0.000000	4349.398717
:	:	:	:
22	2.408169	-39.056546	30.307041
23	2.210029	-38.058223	3.082379
24	2.941625	-37.898186	0.380444

Figure 5 shows the projected ellipsoidal information taken from the three, initial simplex positions.

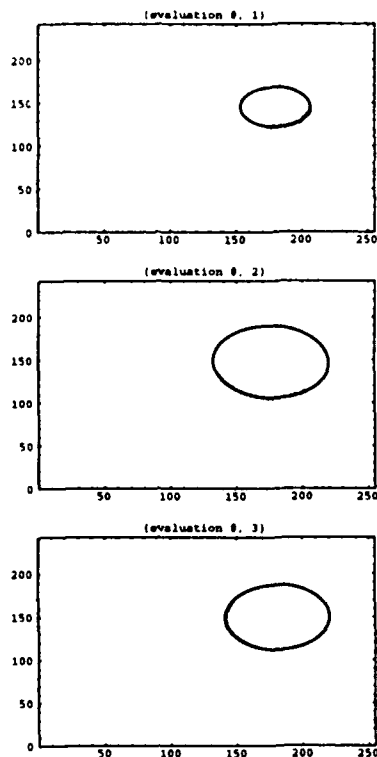


Figure 5: Ellipses recovered by sweeping the initial simplex positions.

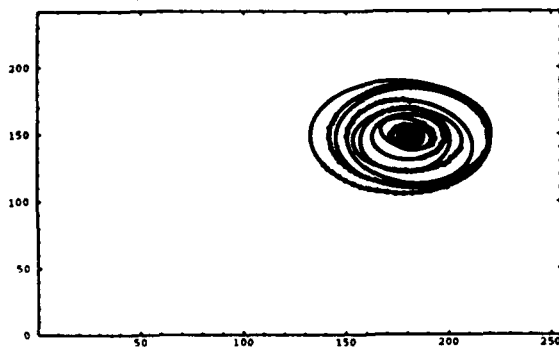


Figure 6: Superimposed ellipses shown for all 25 positions

In the figure, the raw data is displayed as point data while the predicted ellipses are drawn in as solid lines.

Figure 6, we show the ellipses generated by all 25 positions investigated. Note that the system is not guaranteed to be monotonically convergent (in terms of the number of evaluations) but the system is convergent none the less.

The system also can be fooled by ellipses

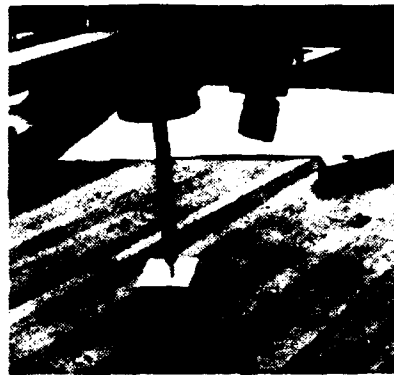


Figure 7: Robot system after performing the insertion task

generated at points which are sampled very close to one another. In the case of the final few ellipses, noise pixels resulted in the oddish ellipsoid calculations. A more intelligent system would detect this condition and would hypothesize about the area of the ellipse taking this into account. But even with noisy data, the system reconstructs an ellipsoid which reflects the general behavior of the points.

In the experiment above and in figure 7, the tracked feature was a 2mm diameter hole. The robot system was able to place the "peg" (a tapered probe) within 3mm of the hole (this using uncalibrated camera data!) In addition, trying the same experiment 3 more times resulted in about the same result, that is: an error of about 3mm for the insertion task. When using a 10mm diameter hole, the robot system almost always succeeds in placing the probe in the hole.

6.1 Evaluations for positions on the initial simplex

Upon closer examination, the modified simplex method does converge as well as a method should taking into account the amount of knowledge we have given it about this system. (See figure 8.) The modified simplex method does suffer from the fault of reexamining points analyzed previously. This can be seen in the overlapping numbers in figure 8. The only way the simplex can shrink itself is by covering all possible

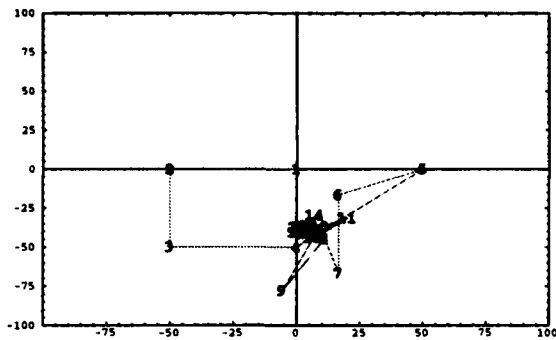


Figure 8: The positions examined by the modified simplex method.

point reflections and then after exhaustively examining all possibilities, it determines the best step for proceeding to the solution state is to contract.

Notice that the simplex method suffers from the fact that it must "overextend" the simplex in all directions before coming to the conclusion that the simplex should be shrunk. This ability allows a simplex to normally "jump" over a local minima and continue its search in a more fruitful valley. In the case of our system, there exists only one minima and simplex need not evaluate all positions when it sees an increase in the area of an ellipse after picking a new point.

This observation brings up several possible places where the algorithm for computing the next position can be improved and makes two insightful observations which are crucial to understanding the alignment problem. The first observation is the fact that we are tracking the centroid of the moving object rather than the true center of the object. In the case where the object is point-like, the center of the object and the centroid of the object are very close together, so the algorithm works. But, in the case of a fairly large object observed through a fairly wide angle lens (take for instance: 12.5mm focal length), the distortion of the center of an object can be significant (on the order of $> 1/2$ the radius of the object) depending on the angle the camera takes with respect to the rotational axis.

The second observation is the fact that by starting with several observations where the rotational axis is far from the object po-

sition, resulting in large ellipses, we can start with accurate estimates of the "family" of ellipses over small rotations. This is in contrast to the smaller ellipses which (because of numerical inaccuracies in estimating the center of the object and problems caused by quantization) need larger movement arcs to adequately recover the parameters of the ellipse. This sweep function is a function of the resolution of the imaging device as well as the size of the object and position of the object.

One possibility for increasing the effectiveness of this process is to use more of the innate properties of the ellipses generated by the process. A more sophisticated search procedure, one based on the physical model of the parabolic surface which is formed by the ellipses areas, would give more satisfactory results.

In addition, productive results will probably be gained from the analysis of other properties of the conic sections. If the component values of the conic sections are traced out as a function of the rotation, the sinusoids generated will show a phase angle difference with respect to the rotation of the end-effector. The magnitude of the sinusoids will determine the net amount of translation of the object with respect to the rotational axis. These four values can probably be used as a control signal to effect a net change to drive all four values to zero which is a position where the sinusoids are both in phase and at zero amplitude with respect to the rotations: the alignment condition. This technique needs to be examined in further detail.

Another problem which must be faced is the problem of small ellipses. When image noise is of the same magnitude as the centroid data the Least Squares fit no longer captures the true centroid information of the object. Remember that the object itself is perspective transformed and the true object center can actually be a great distance from the objects projected center. It may be possible to use the centroid information, if we are able to recover the varying amounts of

skew caused by perspective. It may also be possible to recover the centroidal information by using the parametric description of the object and divining the focal points of the object, the generating lines, and/or the eccentricity of the ellipses.

The final positioning error may be improved by using a set of movement primitive vectors defined by a spiral like the logarithmic spiral or some member of the family of spirals, which can exploit the properties of containment and possibly approach with an incremental goodness-of-fit function (which may be a property of the spiral).

7 CONCLUSIONS

We have demonstrated a method for performing a three dimensional task in essentially two dimensions. The peg-in-hole servoing task and the vernier alignment task both benefit from a method which can constrain the initial position of the object (to a high degree) and which can essentially turn a three dimensional search problem into a two dimensional search in unimodal space. We have presented such a method which converges to a solution state even when using a very simple convergence algorithm.

The key features/contributions of our system:

- It does not require calibrated cameras.
- It converges with simple search algorithms.
- Even with the two constraints above, the alignment results are very good (our experiments have shown that we can position a probe within 3mm of a 2mm feature consistently.)

Active camera motion that recovers image space properties of tracked objects has shown itself to be useful in performing alignment tasks without the need to calibrate the camera systems.

References

- [1] D. Bennett, J. Hollerbach, and D. Geiger. Autonomous robot calibration for hand-eye coordination. In *Robotics Research*, volume 5, pages 137-144. MIT Press, 1989.
- [2] R. M. Haralick. Solving camera parameters from the perspective projection of a parameterized curve. *Pattern Recognit.*, 17(6):637-645, 1984.
- [3] V. Hayward. *RCCL User's Guide*. McGill Research Centre for Intelligent Machines, McGill Univ., Montreal, Quebec, Canada, 1984.
- [4] R. J. Holt and A. N. Netravali. Camera calibration problem: Some new results. *CVGIP: Image Understanding*, 54(3):368-383, Nov. 1991.
- [5] A. Inc. *Programming PIPE Model 1 Systems*. Aspex Inc., 530 Broadway, NYC, NY 10012, 1987.
- [6] H. M. Karara. *Non-Topographic Photogrammetry*. American Society of Photogrammetry and Remote Sensing, Falls Church, VA, 1989.
- [7] M. J. Magee and J. K. Aggarwal. Determining the position of a robot using a single calibration object. In *Proc. IEEE Int. Conf. Robotics and Automat.*, pages 140-149, Mar 1984.
- [8] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 1987.
- [9] R. Safaee-Rad, K. C. Smith, and B. Benhabib. Accurate estimation of elliptical shape parameters from a grey-level image. In *Proc. IEEE Int. Conf. Pattern Recognit.*, pages 20-26, June 1990.
- [10] R. Safaee-Rad, I. Tchoukanov, K. C. Smith, and B. Benhabib. Three-dimensional location estimation of circular features for machine vision. *IEEE Trans. on Robotics and Automation*, 8:624-640, 1992.

- [11] H. S. Sawhney. *Spatial and Temporal Grouping in the Interpretation of Image Motion*. PhD thesis, Univ. of Mass.-Amherst, Feb 1992.
- [12] H. S. Sawhney, J. Oliensis, and A. R. Hanson. Description and reconstruction from image trajectories of rotational motion. In *Proc. 3rd IEEE int. Conf. Comput. Vision*, pages 494-498, Dec. 1990.
- [13] Y. C. Shiu and S. Ahmad. 3d location of circular and spherical features by monocular model-based vision. In *Proc. IEEE Conf on Sys., Man and Cyber.*, pages 576-581, Nov. 1989.
- [14] I. Tchoukanov, R. Safaee-Rad, B. Benhabib, and K. C. Smith. Sub-pixel edge detection for accurate estimation of elliptical shape parameters. In *Proc. CSME Mechan. Eng. Forum*, pages 313-318, June 1990.
- [15] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE J. of Robotics and Automation*, RA-3(4), Aug. 1987.
- [16] R. Y. Tsai. *Synopsis of Recent Progress on Camera Calibration for 3D Machine Vision*. The MIT Press, 1989.
- [17] B. H. Yoshimi and P. K. Allen. Alignment using an uncalibrated camera system. Technical Report TR-002-93, Columbia University, 1993.
- [18] G. S. Young, T. H. Hong, M. Herman, and J. C. S. Yang. Kinematic calibration of an active camera system. Technical Report NISTIR 4715, NIST-Robot Systems Division-Sensory Intelligence Group, Bldg. 220 Rm B124, Gaithersburg, MD, 20899, Nov. 1991.

Performance Evaluation of Multispectral Analysis for Surface Material Classification

Stephen J. Ford
John B. Hampshire II*
David M. McKeown, Jr.

Digital Mapping Laboratory
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891

Department of Electrical and
Computer Engineering *
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

Previous work has shown the feasibility of merging surface material information derived from moderate resolution multispectral imagery with estimates of height based upon stereo matching in high resolution panchromatic imagery. The goal is to use surface material information, normally highly correlated with object location in complex urban scenes, as a source of information for small scale mapping of man-made structures such as buildings and roads, as well as natural features such as soil, vegetation, and water. The fusion of height estimates with surface material estimates provides a unique synthetic three dimensional dataset that is not directly available in any airborne imaging sensor.

The focus of this paper is to present a performance evaluation of two classification techniques, gaussian maximum likelihood and differential radial basis function, on the task of surface material analysis. In order to carry out this evaluation we have created several highly detailed ground truth segmentations based upon manual analysis of the multispectral imagery, as well as by inspection of panchromatic imagery acquired over the same area. Tools built for the generation and validation of the ground-truth surface material map are also discussed.¹

1. Introduction

With the availability of moderate resolution multispectral imagery, comparable in spatial resolution to aerial mapping imagery, opportunities exist to exploit the inherent spectral information of multispectral imagery to aid urban scene analysis for cartographic feature extraction. Moderate resolution multispectral imagery with spatial resolution ranges of 5 to 8 meters can be collected with existing airborne multispectral scanners like Daedalus, AVIRIS, and MEIS.

Our research in multispectral scene information fusion utilizes moderate resolution airborne imagery (8 meter) and high resolution panchromatic aerial photography (1.2 meter). Using traditional spectral classification techniques, surface material information is derived from the multispectral imagery, refined by monocular segmentations from the panchromatic imagery and fused with high resolution stereo disparity maps [Ford and McKeown 92a, Ford and McKeown 92b].

The focus of this paper is to present a performance evaluation of two classification techniques, gaussian maximum likelihood and differential radial basis function, to perform surface material analysis. In order to do this evaluation we have created several highly detailed ground truth segmentations based upon manual analysis of the multispectral imagery, as well as by inspection of panchromatic imagery acquired over the same area. Tools built for the generation and validation of the ground-truth surface material map are also discussed.

In the remainder of this section we give a brief overview of the Daedalus scanner and the surface material classification task.

¹ This research was sponsored by the U.S. Army Topographic Engineering Center and the Defense Advanced Research Projects Agency under Contract DACA72-91-C-0014. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Topographic Engineering Center, or the Defense Advanced Research Projects Agency, or of the United States Government.

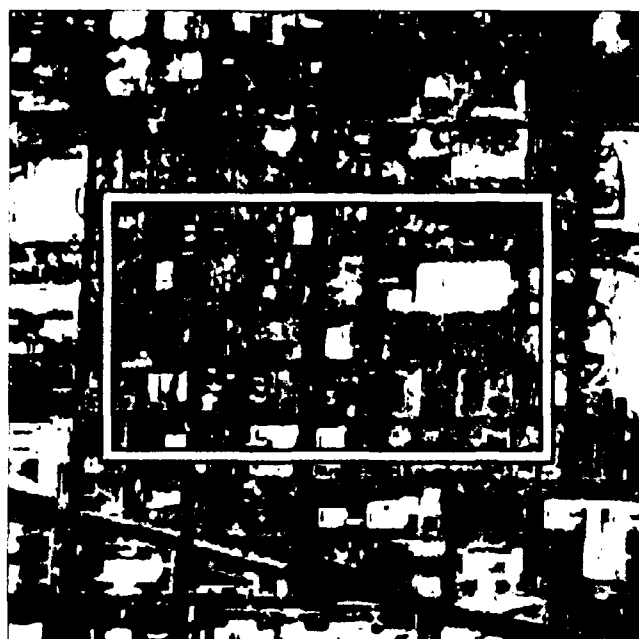


Figure 1: GAOI Site (8 meter)

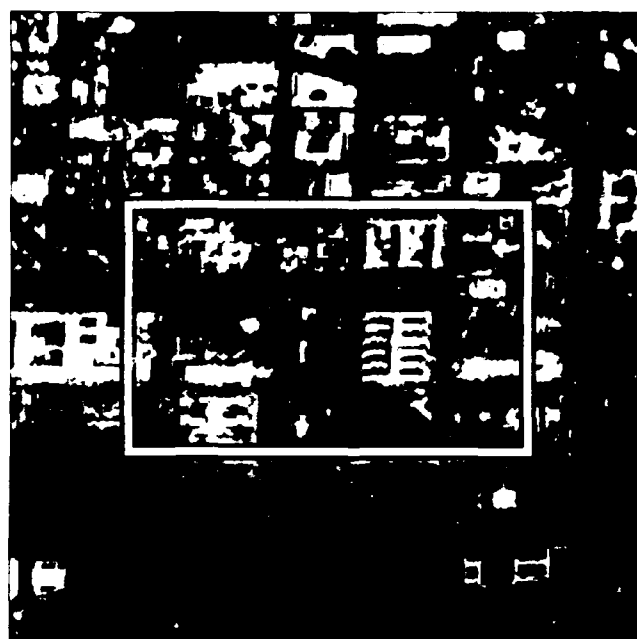


Figure 2: CIVIL Site (8 meter)

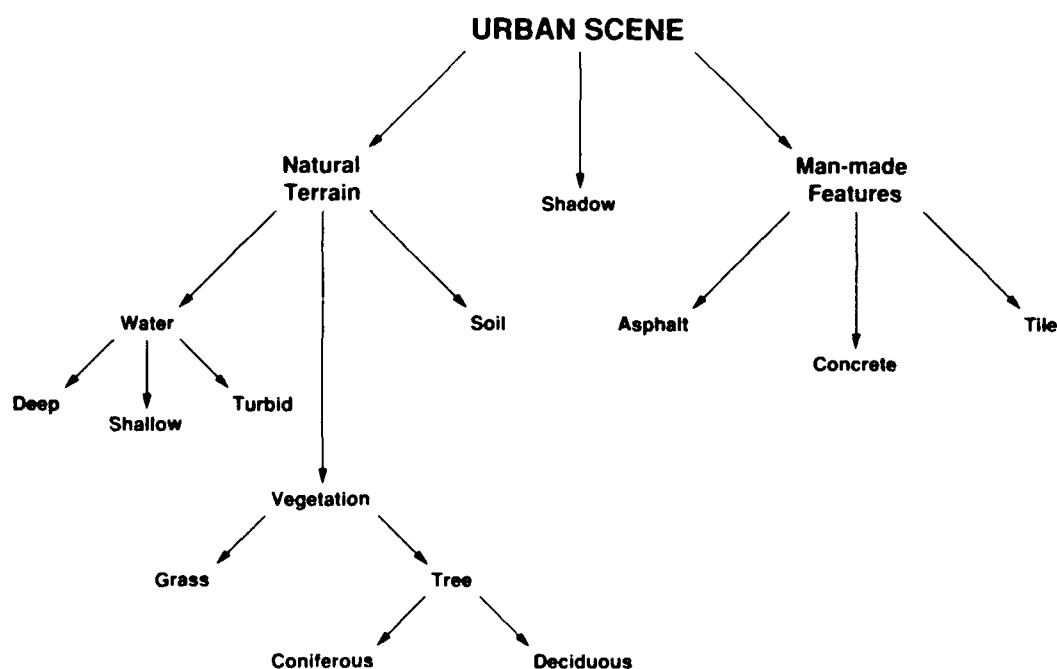


Figure 3: Urban Spectral Classes

1.1. The Daedalus Scanner

The Daedalus Airborne Thematic Mapper (ATM) is an aircraft-based multispectral imaging system. It contains eleven spectral channels, ten of which range from the visible (.4 to .75 micron), to the near infrared (.76 to 1.0), to the shortwave infrared (1.5 to 2.5 micron). The eleventh channel is a thermal band between 8.5 and 13 microns whose physics is fundamentally different than the channels in the reflective part of the electromagnetic spectrum. For more detail on the scanner parameters see [Ford and

McKeown 92a].

By collecting reflected energy in several spectral regions, multispectral imaging systems provide an insight into a material's spectral signature. Using the spectral energy measurements from a set of multispectral bands, individual multispectral image pixels can be assigned or classified into spectral classes (e.g. grass, tree, water, soil, etc.) of similar measurements, based on the multispectral image pixel's intensity or brightness values [Sabins 87].

Since the Daedalus scanner is aircraft-based, the sensor has the capability to acquire moderate resolution multispectral imagery by flying the system at lower altitudes. In the case of our datasets, the scanner was flown to achieve a ground sample distance of approximately 8 meters per pixel. This imagery was collected for the United States SPOT HRV Simulation Campaign [SPOT 83, SPOT 84] and should not be confused with actual SPOT HRV imagery at 20 meter spatial resolution.

1.2. Surface Material Classification

Figures 1 (GAO1) and 2 (CIVIL1) show two of the urban test sites with which we have been experimenting. Both images are visually presented as near infrared images using Daedalus bands 7, 5, and 3.² The outlined regions indicate the area used for classification. The scene content is representative of a complex urban area with buildings, street networks and landscaped areas.

The objective of our classification task involves the generation of surface material classmaps at a *coarse* level for urban multispectral scenes. Coarse level means we are initially only interested in characterizing the primary level of land-cover detail. In our urban analysis problem, the primary land-cover types of most interest to us are water, vegetation, soil and man-made features. In Figure 3, these primary land-cover types are further divided into specific spectral classes based upon visual interpretation of the Daedalus ATM multispectral imagery. The inclusion of a shadow feature in the spectral class hierarchy alleviates misclassifications of shadow pixels as water spectral features due to spectral similarities between the two features.

2. Two Techniques for Multispectral Classification

Traditional multispectral classifiers can be categorized into one of two methods: unsupervised and supervised. The primary distinction between the two multispectral classification procedures centers around the involvement and interaction of the image analyst or domain expert with the classification process. Typically, time must be spent by the image analyst to identify candidate spectral classes, called training sets, prior to supervised classification.

In this section we describe two classification techniques, Gaussian Maximum Likelihood (GML) and Differential Radial Basis Function (DRBF). Both are supervised classifiers requiring training sets to characterize the spectral classes of interest. However, their utilization of training sets for spectral class

characterization and the determination of a multispectral pixel's class assignment differ. We then discuss the generation of training datasets, followed by a discussion of our methodology for ground truth generation for our surface material classes. The generation of a ground truth surface material map is critical for the quantitative analysis of classification results discussed in Sections 3 and 4.

2.1. Gaussian Maximum Likelihood Classifier

In the maximal likelihood classification model, the training set of a spectral class statistically characterizes the class in the form of a mean vector and covariance matrix with the dimensionality being determined by the number of multispectral bands used in the statistical analysis. Sufficient training samples for each spectral class must be present to allow reasonable estimates of the mean vector and covariance matrix [Richards 86, Swain and Davis 78]. The mean vector characterizes average intensity or brightness level for each multispectral band in the spectral class, while the covariance matrix describes the shape and orientation of the population of the spectral class, assuming a multivariate normal distribution. Diagonal entries of the covariance matrix contain the variance or dispersion of the brightness levels for each multispectral band of the spectral class while off-diagonal entries indicate the degree of correlation between a given pair of multispectral bands.

The gaussian maximum likelihood (GML) classifier assumes that the spectral class probabilities are multivariate normal distributions. This is an assumption, rather than a demonstrable property of natural spectral classes [Richards 86]. The probability distribution of each individual spectral class is modeled by using its mean vector and covariance matrix as calculated from its training set. When classifying a multispectral image pixel, the probability of the pixel belonging to each of the candidate spectral classes is determined and assigned to the spectral class with the highest probability.

2.2. Differential Radial Basis Function Classifier

The differential radial basis function (DRBF) classifier is a modified version of the Gaussian Radial Basis Function (RBF) neural network architecture [Broomhead and Lowe 88, Moody and Darken 88, Poggio and Girosi 89, Medgassy 61]. The paradigm is identical to the maximum likelihood model with four major exceptions:

- The DRBF associates a discriminant function with each spectral class rather than associating an unparameterized distribution with each spectral class. As a result, the DRBF employs discriminative learning while the GML uses a probabilistic model.
- The DRBF's discriminant functions have a peak value of unity, whereas the GML discriminant

² Color images have been replaced by their corresponding luminance (Y) component from a RGB to YIQ transformation for black and white reproduction.

functions have unit area, a requirement for probabilistic models.

- The covariance matrix associated with each discriminant function is diagonal, and all of its diagonal elements have the same value (i.e., the covariance matrix has orthonormal eigenvectors and all of its eigenvalues are identical). For this reason, each discriminant function has $C^2 - 1$ fewer parameters than its maximum likelihood counterpart, where C denotes the dimensionality of the spectral intensity vector. In the case of an 11-element spectral intensity vector representing 11 spectral classes, the maximum likelihood classifier has 1452 parameters compared to the DRBF classifier's 132. Thus, the DRBF has an order of magnitude fewer parameters than its maximum likelihood counterpart.

- The model is trained differentially [Hampshire 93] using the classification figure-of-merit (CFM) objective function [Hampshire and Waibel 90], rather than by the method of maximum likelihood. Differential learning via CFM is a discriminative form of learning that focuses on classifying patterns with minimum probability of error. This contrasts with probabilistic learning strategies such as maximum likelihood and conventional neural network learning procedures, which focus on estimating probabilities [Hampshire and Kumar 92].

2.3. Training Data Sets

Block training sets, consisting of homogeneous areas of pure pixels representative of the candidate spectral classes were collected manually from various regions distributed throughout the entire Daedalus ATM multispectral imagery. The candidate spectral classes are listed in Table 1 with the number of training samples or pixels per spectral class for each of the classifiers. One can note that these are relatively small sample sets from the original multispectral imagery over Washington, D.C., measuring 716 rows by 3000 columns, or about 2 million samples in each of the eleven Daedalus bands. These samples were selected prior to the selection of the GAO1 and CIVIL1 test sites, so as to cover a wide range of materials visible over the entire swath from Virginia, across the Potomac river, and through the center of Washington, D.C..

One artifact of our training sample selection was that no attempt was made to acquire a balanced or equally populated set of training data across each of the spectral classes. One property of the DRBF classifier is that it is an effective way of classifying multispectral data with simple models that require relatively small training samples (i.e., training data sets). However, a related property of the DRBF classifier (and differentially trained classifiers in general) is that they require *balanced* training samples in which the number of training examples for each spectral class

Spectral Class	Number of Training Samples	
	GML	DRBF
asphalt	740	740
concrete	720	720
coniferous tree	52	52
deciduous tree	9759	1259
deep water	16433	933
grass	2544	1044
shadow	140	140
shallow water	887	887
soil	354	354
tile	260	260
turbid water	2269	769

Table 1: Spectral Classes Used in Classification

corresponds to the true *a priori* probability of that class.

Each discriminant function of the maximum likelihood model is trained independent of the other functions, and absent any *a priori* knowledge regarding the prior probabilities of each spectral class, the final classification of the maximum likelihood model assumes that these priors are all equal. In this sense, maximum likelihood training is relatively insensitive to unbalanced training samples, assuming the class prior probabilities are roughly equal. Because differential learning requires that all discriminant functions be trained simultaneously, the resulting classifier is quite sensitive to unbalanced training samples. If the empirical probability of a given spectral class does not correspond to the true *a priori* probability of that class, then the training set statistics are not representative of the spectral intensity vector's true probabilistic nature. As a result of unbalanced training samples, the differentially trained classifier tends to form inappropriate decision boundaries, making more classification errors for under-represented spectral classes than a comparable maximum likelihood classifier.

During some preliminary testing using the GML's training data, it was observed that the DRBF classifier has trouble distinguishing shadow regions from water. This is due to two factors, the first and most obvious of which was that the spectral intensities of these ground-truth classes are quite similar. The second factor involved the water to shadow training sample ratio. Because the number of water training examples was over 19,000 and the number of shadow training examples is 140, the training sample ratio of water to

shadow is more than 100:1. However, the ratio of water to shadow in the test samples is less than 1:1. As a result we reduced the training set size for the DRBF classifier to those shown in Table 1. Significant reductions in training set size were also performed for vegetation classes. No special selection criteria were employed to perform this reduction; we simply used the N first training samples for each class in those cases where the GML training populations were large. Tables 2 thru 5 in Section 3 include some of the original DRBF results with the unbalanced training sets labeled as DRBF(1). It can be observed that the balanced training DRBF(2) performed better than the unbalanced experiment DRBF(1).

2.4. Generation of Surface Material Ground Truth

Our previous work in ground truth generation used high resolution black and white aerial photographs. These manual segmentation tools are useful for generating ground truth where geometric relationships, such as building size, shape and boundaries are the primary focus. With regard to spectral classification, the ground truth needs to represent the material types located in the scene. As a consequence, our previously developed manual hand segmentation tools were inadequate and inappropriate. We have addressed this inadequacy by the development of an interactive supervised classification tool, ICLASS, to generate surface material ground truths in the form of surface material ground truth classmaps.

The ground truth generation procedure consists of a two stage process using ICLASS. Initially, a surface material classmap is generated for each of the individual spectral classes visually present in the GAO1 and CIVIL1 test sites. An individual surface material classmap is built by manually segmenting the moderate resolution multispectral image regions containing the surface material of interest using various false color composite presentations. The near infrared color presentation using Daedalus ATM band 7, 5 and 3 has proven to be the most useful visually. When difficulties in visually distinguishing between surface material types using the moderate resolution (8 meter) multispectral imagery occurred, collateral imagery in the form of high resolution panchromatic aerial imagery was referenced in attempts to resolve ambiguities during surface material segmentation with the multispectral imagery. The collateral imagery was helpful in varying degrees due to the difference in scene content between the two image datasets; the aerial imagery was acquired in 1976, while the multispectral imagery was collected in 1983.

With creation of surface material classmaps for each of the individual spectral classes complete, the surface material classmaps are combined together to generate the surface material ground truth classmap. During the segmentation process, it is quite possible that an individual multispectral image pixel may be assigned

to multiple spectral classes, especially along surface material transition boundaries. These conflicting pixels are flagged, reassigned to UNCLASSIFIED and resolved by visually displaying the conflict pixel for assignment. The center column in Figures 4 and 5 shows the resulting surface material ground truth classmaps. Every pixel in the GAO1 (14014 pixels) and CIVIL1 (12180 pixels) test sites were manually labeled.

While the training set data acquired over the entire Daedalus image had examples from each of the eleven surface material classes, not all classes were present in the GAO1 and CIVIL1 sites. Missing from both scenes were soil, coniferous tree, deep water, shallow water, and turbid water classes. A secondary issue, not explored in this paper, is the issue of intrinsic accuracy of the surface material ground truth. A recent study has shown statistically significant differences in ground truth labeling accuracy across multiple analysts using Landsat Thematic Mapper imagery [McGwire 92]. Such imagery has significantly lower spatial resolution than the Daedalus scanner data and that may relate to increased variability in human labeling.

Nevertheless, it is certainly the case that our ground truth class map contains some errors due to our inability to discern precise material boundaries or to label small pixel populations embedded in large areas of nearly homogeneous surface materials. For example, at times it was visually difficult to distinguish between the deciduous tree canopy and the underlying grass areas. A standard deviation classification map may be useful distinguishing between the textured tree canopy and the flat areas of grass. Similarly, it can be very difficult to visually distinguish between asphalt and shadow. The Daedalus thermal band 11 was not extensively used during the manual ground truth segmentation. A re-examination of asphalt and shadow transition areas using the thermal band may be useful in improving the ground truth.

3. Classification Test Results

Surface material classmaps were generated for GAO1 and CIVIL1 using the Gaussian Maximum Likelihood and Differential Radial Basis Function classifiers. The resulting surface material classmaps for the two classifiers are shown as greylevel classification images in the left and right columns of Figures 4 and 5. The manually generated Ground Truth is shown in the center column of each Figure. Each horizontal set of images are side-by-side comparisons of the classification results for each of the major surface material classes: MAN-MADE, SHADOW/SOIL, VEGETATION, and WATER. Within each class, sub-class features are distinguished by grey shades, with white being reserved in all cases to indicate no pixels classified as a member of this class. All results shown in these images represent the classification achieved using all 11 Daedalus spectral bands for training and classification.

GAO1 Site			
Classifier	4 Bands (%)	10 Bands (%)	11 Bands (%)
GML	79.6	85.9	87.8
DRBF (1)	79.1	*	78.1
DRBF (2)	85.4	84.8	85.5
14014 Pixels Sampled			

Table 2: GAO1 Coarse Classification Accuracy

CIVIL1 Site			
Classifier	4 Bands (%)	10 Bands (%)	11 Bands (%)
GML	75.9	78.4	83.2
DRBF (1)	76.8	*	79.9
DRBF (2)	79.9	80.8	84.3
12180 Pixels Sampled			

Table 3: CIVIL1 Coarse Classification Accuracy

GAO1 Site			
Classifier	4 Bands (%)	10 Bands (%)	11 Bands (%)
GML	58.7	65.2	63.6
DRBF (1)	59.0	*	60.1
DRBF (2)	66.8	67.7	68.7
14014 Pixels Sampled			

Table 4: GAO1 Fine Classification Accuracy

CIVIL1 Site			
Classifier	4 Bands (%)	10 Bands (%)	11 Bands (%)
GML	49.1	53.2	55.3
DRBF (1)	58.2	*	62.5
DRBF (2)	63.7	64.6	68.8
12180 Pixels Sampled			

Table 5: CIVIL1 Fine Classification Accuracy

The surface material classmaps generated by the Gaussian Maximum Likelihood and Differential Radial Basis Function classifiers were compared against the surface material ground truth classmaps for GAO1 and CIVIL1. Classification accuracies were performed for both coarse and fine surface material class sets. In the following sections we present overall classification

GAO1 Site			
Classifier	4 Bands (%)	10 Bands (%)	11 Bands (%)
GML	50.9	58.6	61.6
DRBF (1)	47.1	*	46.7
DRBF (2)	62.2	61.7	60.5
14014 Pixels Sampled			

Table 6: GAO1 Coarse Classification Kappa

CIVIL1 Site			
Classifier	4 Bands (%)	10 Bands (%)	11 Bands (%)
GML	58.8	60.4	67.3
DRBF (1)	57.8	*	63.3
DRBF (2)	63.8	65.1	70.1
12180 Pixels Sampled			

Table 7: CIVIL1 Coarse Classification Kappa

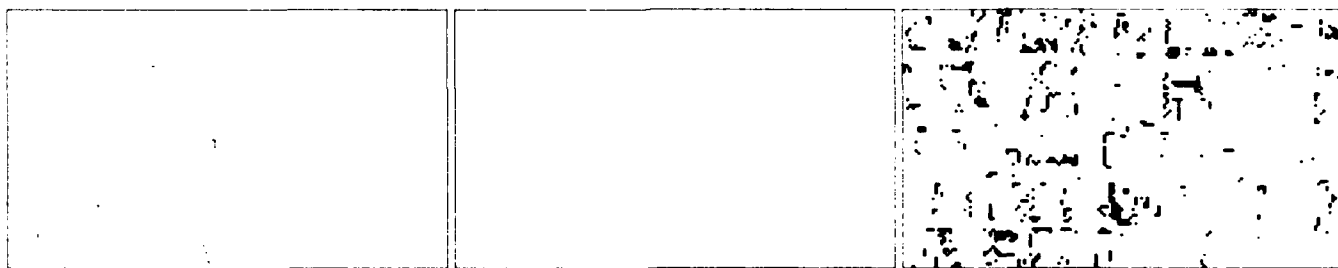
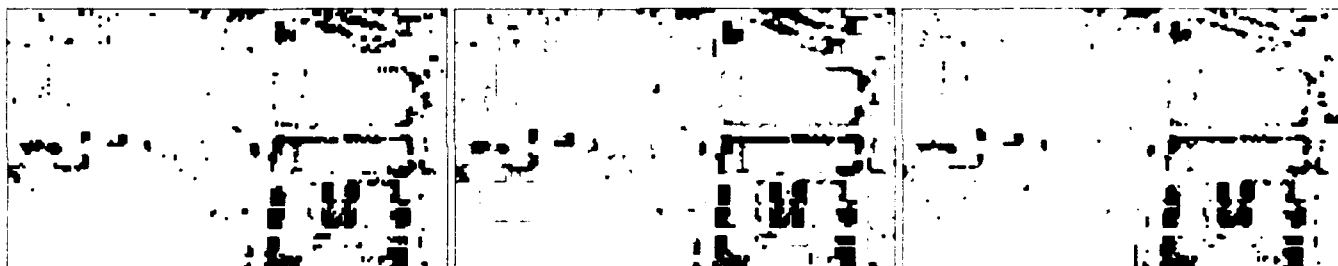
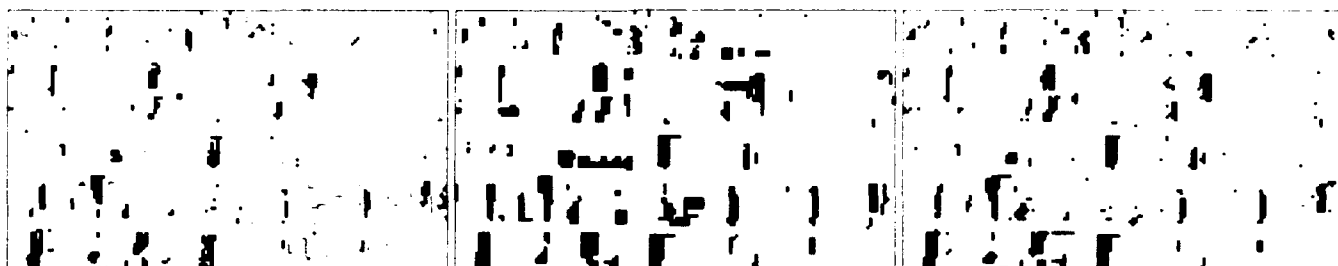
GAO1 Site			
Classifier	4 Bands (%)	10 Bands (%)	11 Bands (%)
GML	44.1	50.3	48.7
DRBF (1)	41.5	*	44.0
DRBF (2)	52.4	53.5	54.3
14014 Pixels Sampled			

Table 8: GAO1 Fine Classification Kappa

CIVIL1 Site			
Classifier	4 Bands (%)	10 Bands (%)	11 Bands (%)
GML	38.0	41.3	44.0
DRBF (1)	46.3	*	51.8
DRBF (2)	53.0	53.9	58.9
12180 Pixels Sampled			

Table 9: CIVIL1 Fine Classification Kappa

accuracies and measure of agreement for each of the methods.

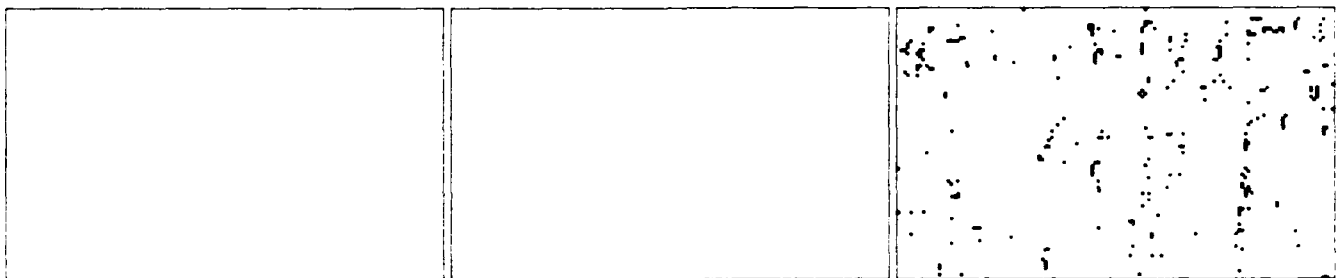
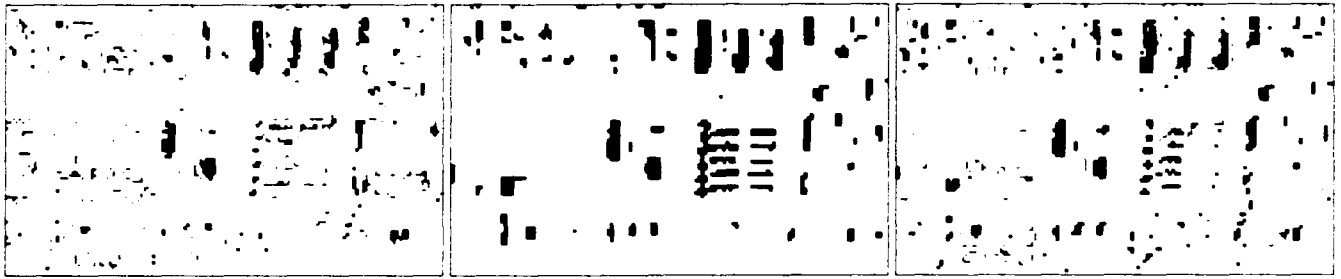
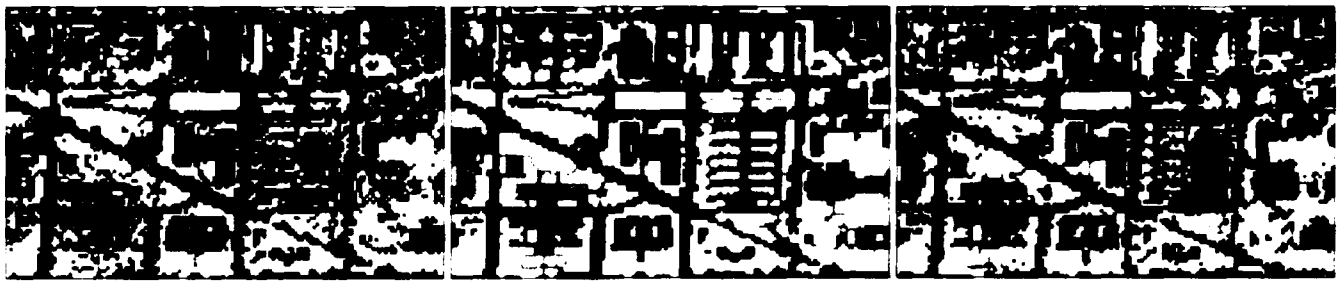


Gaussian Maximum Likelihood

Ground Truth

Differential Radial Basis Function

Figure 4: GAOI Generated Surface Material Classmaps



Gaussian Maximum Likelihood

Ground Truth

Differential Radial Basis Function

Figure 5: CIVIL1 Generated Surface Material Classmaps

3.1. Classification Accuracy

With the generation of classmaps by different classification methods, it is necessary to have some method for evaluating and comparing the accuracy of the generated classmaps to aid in the assessment and improvement of these methods. Various techniques have been developed and implemented in the remote sensing community to determine and evaluate the accuracy of land-use/land-cover maps derived from remotely sensed data [Aronoff 82, Dicks and Lo 90, Fitzpatrick-Lins 81, Story and Congalton 86]. The basic accuracy assessment procedure involves the selection of *samples* from the land-use/land-cover map, verification of the samples using ground truth, and statistical evaluation of the verified samples from the land-use/land-cover map [Congalton, et. al. 83, Congalton, et. al. 84, Congalton and Mead 86].

In our accuracy assessment process, *all* samples (i.e. pixels) contained in the generated surface material classmap are used. This requires the availability of a surface material ground truth classmap with the same coverage as the test area, a condition rarely found in most remote sensing experiments. Our surface material ground truth classmap provides the necessary ground truth during the verification process of the generated surface material classmap.

Each of the Tables 2 through 5 give classification accuracies with respect to our ground truth segmentation. The row labeled GML gives the results for the gaussian maximum likelihood classifier. The rows labeled DRBF(1) and DRBF(2) are the results of the differential radial basis function classifier. As described in Section 2.3, DRBF(1) was an initial experiment using unbalanced training sets and was superseded by the results of the DRBF(2) classifier, using the balanced training sets.

Three experiments were run. The first column shows accuracy using four of the 11 Daedalus spectral bands (3, 5, 7, 10) for classification. These bands were selected by calculating the average Jeffries-Matusita Distance [Mausel, et. al. 90, Richards 86, Swain and Davis 78] using the statistics from the spectral class training sets, in order to rank spectral class separability for all four band combinations.

The second column shows the classification accuracy using all ten of the reflective bands of the Daedalus scanner. The third column gives classification accuracy using all eleven bands, including the thermal band. From a remote sensing standpoint this generally would not be performed, since the physics of the thermal band is quite different than the reflective bands. Nevertheless we tried this experiment and were surprised to see some measurable improvement over the results using the ten reflective bands. These improvements were not only evident in the classification accuracy measure, but also in the measure of agreement, the Kappa coefficient,

discussed in Section 3.2.

The results were tabulated in two ways, first using the coarse classification metric that grouped the eleven surface material classes into five groups: man-made, vegetation, shadow, soil, and water. The groupings are shown in Table 12. The second tabulates accuracy for the fine classification of each of the surface material classes. For both the GAO1 and CIVIL1 sites there appears to be no significant difference for coarse analysis between the GML and DRBF(2) classifiers. Both performed quite well, 84 to 87 percent accuracy, in the 11 band case. Similar performance was achieved in the 10 band case, with noticeably poorer results for the 4 band case. This is interesting since the spectral class separability analysis that led us to select these four bands indicated a high degree of information content. This may well be the case with respect to spectral class separability, but it is clear that the use of additional bands always improved classification accuracy in our experiments.

3.2. Measure of Agreement

An additional metric, introduced to the remote sensing community by Congalton et al. [Congalton, et. al. 83, Congalton, et. al. 84], is also commonly used as a measure of classification accuracy. It is called the Kappa coefficient of agreement and has been used as a standard measure when reporting classification accuracy [Hudson 87, Rosenfeld 86]. Cohen [Cohen 60] developed the Kappa coefficient for nominal scales which measures the relationship of beyond chance agreement to expected disagreement. An advantage of the Kappa coefficient is that its calculation takes into consideration off-diagonal entries of the error matrix, or errors of omission and/or of commission. The Kappa coefficient provides a measure of difference between the observed agreement between two classmaps and agreement that is contributed by chance. It theoretically deflates accuracy statistics based upon chance occurrence of correct classification [Congalton, et. al. 83, Rosenfeld 86, Hudson 87, Dicks and Lo 90].

Tables 6 through 9 show the Kappa accuracies that correspond to the original accuracy analysis in Tables 2 through 5. One can observe that the overall classification accuracies have been reduced significantly in most cases, yet the overall trends in relative performance are maintained. In order to understand the details of the strengths and weaknesses of our classification techniques, we present a more detailed error analysis, in terms of coarse and fine errors of commission and omission, in the following section.

4. Classification Performance Evaluation

In this section we describe a more detailed accuracy assessment procedure to evaluate the surface material classmaps for the GML and the DRBF(2) classifiers. As we have seen in Section 3.1, the most common way

to express the accuracy of a generated classmap is by a statement of the percentage of the classmap that has been correctly classified when compared with a reference classmap or ground truth. In addition to this overview one may desire a more detailed tabulation in the form of an error or confusion matrix. In this kind of tally, the reference classmap (represented by the matrix columns) is compared to the generated or test classmap (represented by the matrix rows). The major diagonal indicates the agreement between these two classmaps. Overall accuracy for a particular generated classmap is then calculated by dividing the sum of the entries that form the major diagonal (i.e. the number of correct classifications) by the total number of samples (i.e. pixels) taken [Story and Congalton 86].

The off-diagonal entries indicate the omission and commission errors. Errors of omission correspond to pixels belonging to the spectral class of interest that the classifier has failed to recognize (false negative), whereas errors of commission are those that correspond to pixels from other spectral classes that the classifier has labeled as belonging to the spectral class of interest (false positive). The former refer to columns of the error matrix, whereas the latter refer to rows [Richards 86].

4.1. Coarse Class Analysis

Due to our objective of generating surface material classmaps at a *coarse* level for urban multispectral scenes, we evaluated the performance of the classification results using surface materials comprised of man-made, vegetation, shadow, soil and water features. For the coarse class analysis, nine of eleven spectral classes are aggregated into three surface materials of man-made, vegetation and water during performance evaluation. Listed in Table 12 are the spectral class membership into the five coarse surface materials along with the Key used in the error matrices. Tables 10 and 11 contain the performance evaluation results for the 11 band GAO1 classification using GML and DRBF(2), respectively, while Tables 13 and 14 highlight the 11 band CIVIL1 classification.

The GML overall classification accuracies are 87.8% and 83.2% for GAO1 and CIVIL1 while the DRBF(2) overall accuracies are 85.5% and 84.3%. Based on the overall accuracies from both sites, the GML and DRBF(2) classification results appear very similar. Upon inspection of the error matrices, man-made and vegetation features account for approximately 90% of the surface materials contained in both sites' ground truth. These features dominate the performance evaluation process when determining the overall classification accuracy which fails to indicate differences between the GML and DRBF(2) classification results.

Further examination of the GAO1 and CIVIL1 error matrices at the coarse classification level show some interesting trends. Referring to Tables 10 through 14,

the GML was able to distinguish between water and man-made or shadow significantly better than the DRBF(2). The GML classified 23 shadow (column C3) pixels as water (row C5) in the GAO1 site and only 1 shadow pixel as water for CIVIL1. In comparison, the DRBF(2) assigned 513 man-made (column C1) and 317 shadow (column C3) pixels as water (row C5) from GAO1 while labeling 196 man-made and 87 shadow pixels for water in CIVIL1. The DRBF(2)'s inability to discriminate water from man-made and shadow was partially influenced by the unbalanced nature of the water training sets originally presented to it as shown in Table 1.

In terms of locating shadow (row C3, column C3) pixels, the DRBF(2) omitted 259 and 249 fewer candidate pixels than the GML in the GAO1 and CIVIL1 sites, respectively. Almost 60% of shadow (column C3) pixels were classified as man-made (row C1) by the GML.

The DRBF(2) also correctly included significantly fewer soil pixels as man-made when compared to the GML for both test sites. Examining the error matrices for CIVIL1, the DRBF(2) classified 348 fewer man-made (column C1) pixels as soil (row C4) than the GML. An example of classifying man-made pixels as soil by the GML is illustrated in Figure 5 along the rooftop of the Department of Interior building in the center-right portion of the image.

At the coarse level, it is not obvious which man-made and water surface material members are contributing to the classification error. Examining the fine classification results against the ground truth at the detailed class level provides more insight into these discrimination errors between individual surface materials.

4.2. Detailed Class Analysis

In the detailed class analysis, all eleven spectral classes are examined as individual surface materials. Tables 15 and 16 contain the performance evaluation results for the 11 band classification of GAO1 and CIVIL1 using the GML and DRBF(2). The GML overall classification accuracies are 63.6% and 55.3% for GAO1 and CIVIL1 while the DRBF(2) overall accuracies are 68.7% and 68.8%. From the overall accuracies, the GML and DRBF(2) performed about the same for GAO1 but are dramatically different for CIVIL1 with the DRBF(2) out-performing the GML.

The sharp drop in GML classifier accuracy between GAO1 and CIVIL1 is due to the major differences in relative scene material composition. The CIVIL1 site is composed of 33% vegetation (i.e. grass and deciduous tree) while the GAO1 site contains 13% vegetation. The ground truth for GAO1 and CIVIL1 in Figures 4 and 5 illustrate the increase in vegetation composition. Referring to the GML error matrix in Table 16, it is clear that the GML has difficulty in distinguishing

Surface Material Ground Truth								
		C1	C2	C3	C4	C5	Row Total	Commission Error
GML Surface Material Test	C1	10648	613	756	0	0	12017	11.4
	C2	12	1205	24	0	0	1241	2.9
	C3	41	2	451	0	0	494	8.7
	C4	201	38	0	0	0	239	100.0
	C5	0	0	23	0	0	23	100.0
	Column Total	10902	1858	1254	0	0	14014	
	Omission Error	2.3	35.1	64.0	*	*		Percent
GML Overall Accuracy = 12304 / 14014 = 87.8%								

Table 10: Error Matrix Showing Results of GAO1 Coarse Classification Using GML

Surface Material Ground Truth								
		C1	C2	C3	C4	C5	Row Total	Commission Error
DRBF(2) Surface Material Test	C1	10176	720	186	0	0	11082	8.2
	C2	85	1089	41	0	0	1215	10.4
	C3	73	8	710	0	0	791	10.2
	C4	55	24	0	0	0	79	100.0
	C5	513	17	317	0	0	847	100.0
	Column Total	10902	1858	1254	0	0	14014	
	Omission Error	6.7	41.4	43.4	*	*		Percent
DRBF(2) Overall Accuracy = 11975 / 14014 = 85.5%								

Table 11: Error Matrix Showing Results of GAO1 Coarse Classification Using DRBF(2)

Key	Surface Material	Members
C1	Man-Made	Asphalt Concrete Tile
C2	Vegetation	Grass Coniferous Tree Deciduous Tree
C3	Shadow	Shadow
C4	Soil	Soil
C5	Water	Deep Water Shallow Water Turbid Water

Table 12: Error Matrix Legend for Coarse Classification

between deciduous trees and grass. It labeled 1560 pixels as grass (row C4) when they were actually deciduous trees (column C3). To the GML, portions of

Surface Material Ground Truth								
		C1	C2	C3	C4	C5	Row Total	Commission Error
GML Surface Material Test	C1	6703	859	502	0	0	8064	16.9
	C2	35	3159	25	0	0	3219	1.9
	C3	22	10	270	0	0	302	10.6
	C4	523	60	11	0	0	594	100.0
	C5	0	0	1	0	0	1	100.0
	Column Total	7283	4088	809	0	0	12180	
	Omission Error	8.0	22.7	66.6	*	*		Percent
GML Overall Accuracy = 10132 / 12180 = 83.2%								

Table 13: Error Matrix Showing Results of CIVIL1 Coarse Classification Using GML

Surface Material Ground Truth								
		C1	C2	C3	C4	C5	Row Total	Commission Error
DRBF(2) Surface Material Test	C1	6740	956	154	0	0	7850	14.1
	C2	117	3011	49	0	0	3177	5.2
	C3	55	34	519	0	0	608	14.6
	C4	175	74	0	0	0	249	100.0
	C5	196	13	87	0	0	296	100.0
	Column Total	7283	4088	809	0	0	12180	
	Omission Error	7.5	26.3	35.8	*	*		Percent
DRBF(2) Overall Accuracy = 10270 / 12180 = 84.3%								

Table 14: Error Matrix Showing Results of CIVIL1 Coarse Classification Using DRBF(2)

the deciduous tree canopy appear spectrally similar to grass while the DRBF(2) is able to discriminate between the two vegetated surface materials. In the coarse analysis, this error was not observed due to the grouping of grass and deciduous tree into vegetation.

We noted in the coarse class analysis that the DRBF(2) had difficulty in distinguishing between water and man-made or shadow. From the error matrices for the fine classification, the DRBF(2) is labeling asphalt (column C1) and shadow (column C5) pixels as deep water (row C9) and turbid water (row C11). As previously stated, it is believed that a more balanced training set would alleviate these errors.

The GML's higher number of soil pixels, as observed in the coarse analysis, is related to the GML's misclassification of concrete (column C2) pixels as soil (row C6). The spectral characterization of soil and concrete is evidently too similar under the GML model.

In general, both classifiers had difficulty in

Surface Material Ground Truth														
		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	Row Total	Commission Error
GML Surface Material Test	C1	4637	873	472	70	152	0	18	0	0	0	0	6222	25.5
	C2	658	2936	0	0	8	0	1	0	0	0	0	3603	18.5
	C3	0	0	64	1	15	0	0	0	0	0	0	80	20.0
	C4	5	7	630	487	5	0	0	0	0	0	0	1134	57.1
	C5	40	1	2	0	451	0	0	0	0	0	0	494	8.7
	C6	43	155	28	10	0	0	3	0	0	0	0	239	100.0
	C7	972	221	71	0	596	0	332	0	0	0	0	2192	84.9
	C8	0	0	23	0	4	0	0	0	0	0	0	27	100.0
	C9	0	0	0	0	0	0	0	0	0	0	0	0	*
	C10	0	0	0	0	23	0	0	0	0	0	0	23	100.0
	C11	0	0	0	0	0	0	0	0	0	0	0	0	*
	Column Total	6355	4193	1290	568	1254	0	354	0	0	0	0	14014	
	Omission Error	27.0	30.0	95.0	14.3	64.0	*	6.2	*	*	*	*		Percent
GML Overall Accuracy = 8907 / 14014 = 63.6%														

Key	Surface Material	Key	Surface Material	Key	Surface Material	Key	Surface Material
C1	Asphalt	C4	Grass	C7	Tile	C10	Shallow Water
C2	Concrete	C5	Shadow	C8	Coniferous Tree	C11	Turbid Water
C3	Deciduous Tree	C6	Soil	C9	Deep Water		

Surface Material Ground Truth														
		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	Row Total	Commission Error
DRBF(2) Surface Material Test	C1	4988	1069	642	69	164	0	86	0	0	0	0	7018	28.9
	C2	674	2896	3	0	4	0	2	0	0	0	0	3579	19.1
	C3	15	0	327	38	33	0	0	0	0	0	0	413	20.8
	C4	0	17	265	447	0	0	0	0	0	0	0	729	38.7
	C5	69	1	8	0	710	0	3	0	0	0	0	791	10.2
	C6	2	53	12	12	0	0	0	0	0	0	0	79	100.0
	C7	128	75	4	2	18	0	258	0	0	0	0	485	46.8
	C8	5	48	12	0	8	0	0	0	0	0	0	73	100.0
	C9	418	5	17	0	274	0	5	0	0	0	0	719	100.0
	C10	0	0	0	0	0	0	0	0	0	0	0	0	*
	C11	56	29	0	0	43	0	0	0	0	0	0	128	100.0
	Column Total	6355	4193	1290	568	1254	0	354	0	0	0	0	14014	
	Omission Error	21.5	30.9	74.7	21.3	43.4	*	27.1	*	*	*	*		Percent
DRBF(2) Overall Accuracy = 9626 / 14014 = 68.7%														

Table 15: Error Matrices Showing Results of GAOI Fine Classification Using GML and DRBF(2)

Surface Material Ground Truth														
		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	Row Total	Commission Error
GML Surface Material Test	C1	3201	409	516	216	37	0	2	0	0	0	0	4381	26.9
	C2	311	1734	0	0	4	0	1	0	0	0	0	2050	15.4
	C3	1	0	572	25	20	0	0	0	0	0	0	618	7.4
	C4	29	4	1560	905	3	0	1	0	0	0	0	2502	63.8
	C5	22	0	10	0	270	0	0	0	0	0	0	302	10.6
	C6	77	443	55	5	11	0	3	0	0	0	0	594	100.0
	C7	664	325	120	7	461	0	56	0	0	0	0	1633	96.6
	C8	0	0	94	3	2	0	0	0	0	0	0	99	100.0
	C9	0	0	0	0	0	0	0	0	0	0	0	0	*
	C10	0	0	0	0	1	0	0	0	0	0	0	1	100.0
	C11	0	0	0	0	0	0	0	0	0	0	0	0	
	Column Total	4305	2915	2927	1161	809	0	63	0	0	0	0	12180	
	Omission Error	25.6	40.5	80.5	22.0	66.6	*	11.1	*	*	*	*		Percent
GML Overall Accuracy = 6738 / 12180 = 55.3%														

Key	Surface Material	Key	Surface Material	Key	Surface Material	Key	Surface Material
C1	Asphalt	C4	Grass	C7	Tile	C10	Shallow Water
C2	Concrete	C5	Shadow	C8	Coniferous Tree	C11	Turbid Water
C3	Deciduous Tree	C6	Soil	C9	Deep Water		

Surface Material Ground Truth														
		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	Row Total	Commission Error
DRBF(2) Surface Material Test	C1	3554	381	705	229	60	0	16	0	0	0	0	4945	28.1
	C2	347	2109	5	4	5	0	6	0	0	0	0	2476	14.8
	C3	29	4	1474	190	32	0	0	0	0	0	0	1729	14.7
	C4	2	10	631	682	0	0	1	0	0	0	0	1326	48.6
	C5	54	1	34	0	519	0	0	0	0	0	0	608	14.6
	C6	2	173	29	45	0	0	0	0	0	0	0	249	100.0
	C7	126	161	13	0	89	0	40	0	0	0	0	429	90.7
	C8	18	53	23	11	17	0	0	0	0	0	0	122	100.0
	C9	107	3	13	0	45	0	0	0	0	0	0	168	100.0
	C10	0	0	0	0	0	0	0	0	0	0	0	0	*
	C11	66	20	0	0	42	0	0	0	0	0	0	128	100.0
	Column Total	4305	2915	2927	1161	809	0	63	0	0	0	0	12180	
	Omission Error	17.4	27.7	49.6	41.3	35.8	*	36.5	*	*	*	*		Percent
DRBF(2) Overall Accuracy = 8378 / 12180 = 68.8%														

Table 16: Error Matrices Showing Results of CIVIL1 Fine Classification Using GML and DRBF(2)

discriminating asphalt (row C1) from deciduous tree (column 3) and grass (column 4). These pixels occur in transition boundaries between streets and overhanging tree canopies or lawn areas, resulting in a mixed pixel. Mixed pixels are a problem commonly encountered in multispectral classification, which is usually rectified as a post-processing step. We have not addressed these issues to date in our research.

5. Conclusions

The goal of this paper was to present a performance evaluation of two classification techniques, gaussian maximum likelihood (GML) and differential radial basis function (DRBF). This analysis was performed using two urban test sites in Washington D.C. for which we have high resolution mapping photography. What we have found is that the GML and DRBF classifiers appear to perform at similar accuracy rates for coarse surface material classification tasks. For a more detailed surface material classification task, the DRBF performed significantly better than the GML when the amount of grass and deciduous tree scene composition increased. The DRBF was able to distinguish between deciduous tree and grass more reliably than the GML as demonstrated by the CIVIL1 test site. We have not explored the extent to which the two methods provide consistent estimates of surface materials, and this may lead to a combined or hybrid analysis technique.

We have also shown that when evaluating classification accuracy, the overall accuracy can be misleading if a dominant feature is present. Consultation of the error matrix should be part of the performance evaluation process to gain a better understanding of the classification results.

At the present time, all pixels contained in the test site are used during performance evaluation. In the coarse class analysis, it was seen that a dominant surface material can bias the overall classification accuracy. Alternative sampling schemes for determining classification accuracy to remove bias of dominant scene features will be investigated.

We plan to continue our research in the fusion of stereo and monocular cues with surface material classification to improve detection and delineation of buildings, roads, and other man-made structures. Our previous research has shown, our ability to co-register stereo disparity data and surface material classmaps. With a firm basis for the accuracy of our material classifications, we can begin to focus on higher level analysis towards object detection, delineation, and recognition.

Acknowledgements

We thank SPOT Image Corporation for supplying the Daedalus ATM multispectral imagery flown in the 1983 SPOT Simulation Campaign. Tom Ory of

Daedalus Enterprises, Inc. was very helpful in supplying information concerning the Daedalus sensor characteristics and image format.

We thank members of the Digital Mapping Laboratory for providing an interesting research environment. Particular thanks go to Ed Allard for software support for data exchange and evaluation analysis, and Jeffrey Shufelt for valuable comments on an earlier draft of this paper.

References

- [Aronoff 82]
Aronoff, S. Classification Accuracy: A User Approach. *Photogrammetric Engineering and Remote Sensing* 48(8):1299-1307, August, 1982.
- [Broomhead and Lowe 88]
D. S. Broomhead and D. Lowe. Multivariable Function Interpolation and Adaptive Networks. *Complex Systems* 2:321-355, 1988.
- [Cohen 60]
Cohen, J. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20(1):37-46, 1960.
- [Congalton and Mead 86]
Congalton, R. and Mead, R. A Review of Three Discrete Multivariate Analysis Techniques Used in Assessing the Accuracy of Remotely Sensed Data from Error Matrices. *IEEE Transactions on Geoscience and Remote Sensing* GE-24(1):169-174, January, 1986.
- [Congalton, et. al. 83]
Congalton, R., Oderwald, R., and Mead, R. Assessing Landsat Classification Accuracy Using Discrete Multivariate Analysis Statistical Techniques. *Photogrammetric Engineering and Remote Sensing* 49(12):1671-1678, December, 1983.
- [Congalton, et. al. 84]
Congalton, R., Oderwald, R., and Mead, R. Erratum: Assessing Landsat Classification Accuracy Using Discrete Multivariate Analysis Statistical Techniques. *Photogrammetric Engineering and Remote Sensing* 50(10):1477, October, 1984.
- [Dicks and Lo 90]
Dicks, S. E. and Lo, T. H. C. Evaluation of Thematic Map Accuracy in a Land-Use and Land-Cover Mapping Program. *Photogrammetric Engineering and Remote Sensing* 56(9):1247-1252, September, 1990.
- [Fitzpatrick-Lins 81]
Fitzpatrick-Lins, K. Comparison of Sampling Procedures and Data Analysis for a Land-Use and Land-Cover Map. *Photogrammetric Engineering and Remote Sensing* 47(3):343-351, March, 1981.

- [Ford and McKeown 92a]
Ford, S. J., and McKeown, D. M. Utilization of Multispectral Imagery for Cartographic Feature Extraction. In *Proceedings of the DARPA Image Understanding Workshop*, pages 805-820. Morgan Kaufmann Publishers, Inc., San Mateo, CA, January, 1992.
- [Ford and McKeown 92b]
Ford, S. J., and McKeown, D. M. Information Fusion of Multispectral Imagery for Cartographic Feature Extraction. In *Commission VII: Interpretation of Photographic and Remote Sensing Data*. Washington, DC, XVII ISPRS Congress, August 2-14, 1992.
- [Hampshire 93]
J. B. Hampshire II. *A Differential Theory of Learning for Efficient Statistical Pattern Recognition*. PhD thesis, Carnegie Mellon University, Department of Electrical and Computer Engineering, expected April, 1993.
- [Hampshire and Kumar 92]
J. B. Hampshire II and B. V. K. Vijaya Kumar. Why Error Measures are Sub-Optimal for Training Neural Network Pattern Classifiers. In *IEEE Proceedings of the 1992 International Joint Conference on Neural Networks*, Vol. 4, pages 220-227. June, 1992.
- [Hampshire and Waibel 90]
J. B. Hampshire II and A. H. Waibel. A Novel Objective Function for Improved Phoneme Recognition Using Time-Delayed Neural Networks. *IEEE Transactions on Neural Networks* 1(2):216-228, June, 1990.
- [Hudson 87]
Hudson, W. D. and Ramm, C. W. Correct Formulation of the Kappa Coefficient of Agreement. *Photogrammetric Engineering and Remote Sensing* 53(4):421-422, April, 1987.
- [Mausel, et. al. 90]
Mausel, P. W., Kramber, W. J., and Lee, J. K. Optimum Band Selection for Supervised Classification of Multispectral Data. *Photogrammetric Engineering and Remote Sensing* 56(1):55-60, January, 1990.
- [McGwire 92]
McGwire, K. C. Analyst Variability in Labeling of Unsupervised Classifications. *Photogrammetric Engineering and Remote Sensing* 58(12):1673-1677, December, 1992.
- [Medgassy 61]
P. Medgassy. *Decomposition of Superposition of Distribution Functions*. Publishing House of the Hungarian Academy of Sciences, Budapest, 1961.
- [Moody and Darken 88]
J. Moody and C. Darken. Learning with Localised Receptive Fields. In Touretzky, Hinton and Sejnowski (editor), *Proceedings of the 1988 Connectionist Models Summer School*. Morgan Kaufmann, San Mateo, CA, 1988.
- [Poggio and Girosi 89]
T. Poggio and F. Girosi. *A Theory of Networks for Approximation and Learning*. AI Memo 1140, MIT, 1989.
- [Richards 86]
Richards, J. A. *Remote Sensing Digital Image Analysis: An Introduction*. Springer-Verlag, Berlin, 1986.
- [Rosenfeld 86]
Rosenfeld, G. H. and Fitzpatrick-Lins, K. A Coefficient of Agreement as a Measure of Thematic Classification Accuracy. *Photogrammetric Engineering and Remote Sensing* 52(2):223-227, February, 1986.
- [Sabins 87]
Sabins, F. F. *Remote Sensing: Principles and Interpretation*. W. H. Freeman, New York, NY, 1987.
- [SPOT 83]
SPOT IMAGE Corporation. *1983 U.S. SPOT Simulation Campaign Auxilliary Information Package* SPOT IMAGE Corporation, 1150 17th Street NW Suite 307, Washington, DC 20036, 1983.
- [SPOT 84]
SPOT Image Corporation. *SPOT Simulation Applications Handbook*. American Society for Photogrammetry, Falls Church, VA, 1984.
- [Story and Congalton 86]
Story, M. and Congalton, R. Accuracy Assessment: A User's Perspective. *Photogrammetric Engineering and Remote Sensing* 53(3):397-399, March, 1986.
- [Swain and Davis 78]
Swain, P. H. and Davis, S. M. *Remote Sensing: The Quantitative Approach*. McGraw-Hill, New York, NY, 1978.

Incorporating Vanishing Point Geometry Into a Building Extraction System

**J. Chris McGlone
Jefferey A. Shufelt**

Digital Mapping Laboratory
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891

Abstract

Knowledge about the imaging geometry and acquisition parameters provides useful geometric constraints for the analysis and extraction of man-made features in aerial imagery, particularly in oblique views. In this paper, we discuss the application of vanishing points for the identification of vertical and horizontal lines, and the use of multiple views for verification of these lines. The vertical and horizontal attributions are used to constrain the set of possible building hypotheses. Preliminary results exploiting these attributions are described.¹

1. Introduction

Building extraction is a fundamental problem in automated cartography [Nicolin 87, Huertas 88, Mohan 89, Irvin 89, Liow 90, McKeown 90, Shufelt 93]. Systems implemented to date have had basic similarities: all have used vertical aerial imagery, assuming simplified imaging geometry in their calculations, and all have used intensity features as the basic cues for feature extraction. Several have made use of shadow geometry for hypothesis generation and verification. Low level boundary determination is usually region-based or based upon geometric analysis of lines found in the image.

Many of these techniques exhibit poor performance when building structures are composed of complex shapes, when there is poor contrast between object and background, and when viewing geometry, building height, and building density cause occlusions and partial views, or views of surfaces other than the building roof. As a result, even in the case of nominally nadir imagery, the three-dimensional nature of the world can not be ignored. In the case of non-traditional mapping photography, particularly oblique views used in aerial photo-interpretation, there is a greater need to explicitly model the viewing geometry; such modeling needs to be performed within the context of a rigorous photogrammetric calculation in order to take advantage of all geometric information available.

Our current experiments have been focused on the modification of BABE (Builtup Area Building Extraction), a building detection system built at CMU [McKeown 90] based on a line-corner analysis method. We have been experimenting with the inclusion of geometric constraints derived from knowledge of the full camera position and orientation. In brief, BABE proceeds through four major phases to incrementally generate building hypotheses. The first phase constructs corners from lines, under the assumption that buildings can be modeled by straight line segments linked by (nearly) right-angled corners. The second phase constructs chains of edges which are linked by corners, to serve as partial structural hypotheses. The third phase uses these line-corner structures to hypothesize boxes, parallelopipeds which may delineate man-made features in the scene. The fourth phase evaluates the boxes in terms of size and line intensity constraints, and the best boxes for each chain are kept, subject to shadow intensity

¹ This work was sponsored by the Defense Advanced Research Projects Agency under Contract DACA76-92-C-0036. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

constraints similar to those proposed in [Nicolin 87] and [Huertas 88]. In addition, the results of the third phase of analysis are directly used as sources of building hypotheses for other modules that perform grouping, shadow analysis, and stereo matching.

Our initial modifications to the BABE system include the use of a rigorous photogrammetric camera model, the incorporation of vanishing point geometry as an additional input to the building hypothesis construction process, and the substitution of exact metric calculations for distances and angles instead of approximations based upon image scale and near-nadir orientation. This paper describes the current status of the BABE system, starting with an overview of vanishing point geometry as used for the extraction of horizontal and vertical edges and a brief description of the BABE system. The current integration of the vanishing point information into BABE is outlined and some preliminary results are given.

2. Vanishing point geometry

As is well known from projective geometry [Barnard 83], parallel lines in a scene meet in a common point in an image of the scene. This point is known as the vanishing point, since it is the image of a point at infinity. In an aerial image vertical lines in the scene meet at the vertical vanishing point, traditionally referred to as the nadir point because it is directly below the perspective center of the image. Sets of parallel lines at varying orientations in a plane have vanishing points which lie along a straight line in the image. If the sets of parallel lines are horizontal, the line is the true horizon.

This apparent convergence of parallel lines gives important cues to the orientation of the image and to the structure of objects within the scene. Previous work has looked at using vanishing points to determine image orientation [Barnard 83] and to determine the structure of objects within the scene [Brillault 92, Lebegue 92].

Most previous work using vanishing point geometry has been done with robotics imagery from standard video cameras viewing objects at close range. The applicability of vanishing point analysis is obvious; perspective effects are strong due to the wide angle lenses, close objects, and

often oblique viewing angles. Image edges corresponding to hallways, doors, and structures are numerous, long and usually have high contrast, allowing good solutions for vanishing points and image orientations.

Aerial imagery presents different problems. The standard vertical viewpoint lessens perspective effects, while individual objects cover a much smaller proportion of the image. Vertical lines in particular are less prominent, typically only a few pixels long. Edge contrast may be lessened due to illumination and atmospheric conditions. It is well known that standard edge detectors have problems extracting such short, weak edges, often distorting their geometry or mistakenly combining them with intersecting edges.

Further, in cartographic applications it is assumed that the aircraft position and orientation in space is fairly well known, and camera properties such as focal length, distortion and sensor type, film, scanning array, etc., are quite well modeled. For these reasons, our approach starts with the assumption that the orientation of the aerial image is known beforehand. Instead of using the vanishing points to determine image orientation, we focus on using the vanishing point geometry to assist in extracting buildings. Of course, given strong enough vanishing point information from the image the orientation can be refined, but in this work no refinement was attempted.

This section outlines the calculation of the vertical vanishing point and the horizon line, and the identification of vertical and horizontal lines using this information.

2.1. Calculation of the vertical vanishing point

The image orientation is specified by a 3 by 3 orientation matrix M which rotates the ground coordinate system into the image coordinate system. This matrix is determined by three independent orientation angles or parameters, e.g., roll, pitch, and yaw [Slama 80].

The vertical vector in object space v_o is $[0, 0, 1]$ and is transformed into the image coordinate system by multiplication with the ground-to-image orientation matrix M .

$$v_i = M v_o$$



Figure 1: Fort Hood test area RADT9WOB.

When the vector v_i is placed at the perspective center of the image (coordinates $0,0,f$), it pierces the image plane $z = 0$ at

$$x = \frac{m_{13}}{m_{33}} f$$

$$y = \frac{m_{23}}{m_{33}} f$$

Since this vector is vertical it is parallel to all other vertical lines in the scene and its image must pass through the vertical vanishing point. However, its image, where the vector pierces the image plane, is

only a point; its image must therefore be the vertical vanishing point.

2.2. Horizontal vanishing point determination

The horizontal vanishing points are calculated using a variant of the Gaussian sphere technique first applied in [Barnard 83].

The Gaussian sphere represents a vector orientation in 3-space as a point on the sphere. As in [Barnard 83] we assume that the perspective center of the image is at the center of the sphere.

the origin of the image coordinate system, and the image is tangent to the sphere.

An "interpretation plane" is associated with each line in the image, passing through the image line and the perspective center. The interpretation plane is represented on the sphere by the point corresponding to the orientation of its normal and, in a dual sense, by the great circle formed by the intersection of the plane and sphere.

The great circles cut by the interpretation planes corresponding to parallel lines in the scene intersect on the sphere in the vanishing point for that set of parallel lines. The vanishing points for sets of parallel lines of different orientations in a plane lie on the vanishing line for that plane — i.e., for horizontal lines, on the horizon line.

Using the known image orientation we calculate the horizon plane and its corresponding great circle on the sphere, the vanishing line for horizontal lines. (The normal to the horizontal plane is, of course, the vertical vector.) The interpretation plane corresponding to each horizontal line in the scene will intersect the horizon line at the vanishing point for horizontal lines in that direction.

To identify parallel sets of horizontal lines, the great circle for each line in the image is formed and intersected with the horizon. The horizon great circle is divided into equal bins along its arc length (instead of quantizing azimuth or elevation) and the number of intersections within each bin is tallied. The bin with the maximum number of intersections corresponds to the most numerous set of parallel lines within the scene.

3. Identification of vertical and horizontal lines

Given the background for the determination of the vertical vanishing point and our method for determining parallel sets of horizontal lines, we proceed with a demonstration and discussion of current performance using an oblique image of a barracks area over Fort Hood, Texas.

3.1. Vertical lines

In order to find vertical lines in the scene each edge in the image is fit to a line constrained to pass through the vanishing point, leaving only the slope of the line to be determined. The residuals are

calculated and if the rms error exceeds 2.0 pixels, the edge is eliminated. Since extremely short edges will have small residuals for any orientation of line fit, edges below a minimum length are eliminated.



Figure 2: Edges for test area RADT9WOB.

The same resection that produces the image orientation used to calculate the vertical vanishing point also calculates the precision of the orientation angles, from which the precision of the vanishing point location can be determined and used to set the acceptance criteria for slopes and line fitting. For oblique imagery, where the vanishing point is usually outside the image area itself, the precision has a small effect. For vertical images, however, the vertical vanishing point is near the center of the frame and is close to the edges being tested. Error in its location can change the slope of the test line significantly and should be taken into account in the line fitting procedure.

As a further test a line not constrained to pass through the vanishing point is also fitted to accepted edges and the slope of that line compared to the direction from the centroid of the edge to the vanishing point. If the slopes do not agree within an angular tolerance of 0.2 radians, the line is eliminated.



Figure 3: Horizontal edges.

3.2. Horizontal edge extraction

Our goal is to identify man-made structures, which are usually defined by perpendicular sets of parallel lines. We therefore examine the histogram bins (as described in section 2.2) and, instead of choosing the single bin with the maximum score, we add the score of each bin to the scores of the bins representing directions perpendicular to it. The maximum of this sum indicates the directions of the strongest mutually perpendicular sets of parallel lines in the scene. In areas where buildings and roads are all on a common grid, this is sufficient; in areas where not all the buildings are parallel to each other, secondary maxima can be determined to label buildings. Bins corresponding to perpendicular directions can be easily determined, since right angles at the center of the sphere between points on the horizon great circle correspond to true right angles between horizontal lines in object space.

Figure 1 shows an oblique image of a barracks area within Fort Hood, Texas. Such scenes are not atypical for military bases or, with some architectural modifications, for houses in a suburban development. Figure 2 shows the edges extracted by an implementation of the Nevatia-Babu line finder [Nevatia 80], while candidate horizontal and vertical edges are shown in Figures 3 and 4. Some edges are labeled as both horizontal and vertical due to the viewing angle of the image, which happened to align many of the horizontal edges with the vertical vanishing point. In such



Figure 4: Vertical edges.

ambiguous cases, external information or other views must be used to decide between these labels.

4. Horizontal and vertical line verification

Given a single view and only geometric information, the inherent ambiguities of perspective projection prevent an absolute determination of whether a given line is horizontal or vertical. False positive identifications due to accidental alignments are unavoidable. Since these false positives increase the number of edges flagged for later analysis and the computational effort required, we would like to eliminate as many as possible.

A first step is filtering against a minimum length or height threshold. Highly textured areas produce a large number of short, randomly oriented edges, some of which will align with the vanishing point of interest. Using the assumed horizontal or vertical orientation for the line, we can calculate an approximate length or height and compare it to the minimum values we would expect to see. For example, if we are looking for buildings, heights will typically be greater than 3 meters and lengths greater than 10 meters. Such constraints can be easily modified by world knowledge to search for a specific set of buildings within a range of heights or volumes. Currently we view this process as one of filtering rather than selection. Each edge segment that passes these filters is given an attribution as either horizontal or vertical. The entire collection of edges can then be used in a

variety of ways to construct plausible building hypotheses. In the following section we describe the use of attributed edge segments to detect and construct possible building corners.

If multiple views of the scene are available, we can use the epipolar condition to determine if consistent edges appear in both images. For each edge in the image, we calculate the epipolar plane through its midpoint and determine which edges, if any, are intersected by the epipolar line on the other image. For lines which lie on corresponding epipolar lines, we can also compare their calculated dimensions. Horizontal lines can also be matched using their calculated directions in object space, obtained from the horizontal line extraction procedure described above.

5. Corner detection with line attributions

The vanishing-point geometry of a scene can provide important additional cues for feature extraction. Under the assumption that man-made features in aerial photography can be modeled by parallelopipeds joined at edges, horizontal and vertical edge segment attributions are useful cues in assembling building hypotheses. We illustrate the utility of these attributions in the context of a building extraction system, BABE, originally designed for analysis of mapping photography having nadir and near-nadir acquisition geometries.

BABE begins processing by generating intensity edges for an image, using a Nevatia-Babu edge finder [Nevatia 80]. BABE applies a range search to locate and connect collinear edges whose endpoints are in close proximity, to address the possibility of fragmented edges. These edges are then used as the basis for corner detection.

BABE performs another range search on the edges, to locate edges which meet at approximately right angles. The intersections of these edges represent the corner points. BABE then uses these corner points to link sequences of edges such that the direction of rotation along a sequence is either clockwise or counterclockwise, but not both, since building structure is assumed to be well modeled by parallelopipeds.

Even when a building can be modeled perfectly by a rectangle, the chain of edges representing it may not be a closed structure, due to extraneous or missing corners in the chain. BABE addresses this

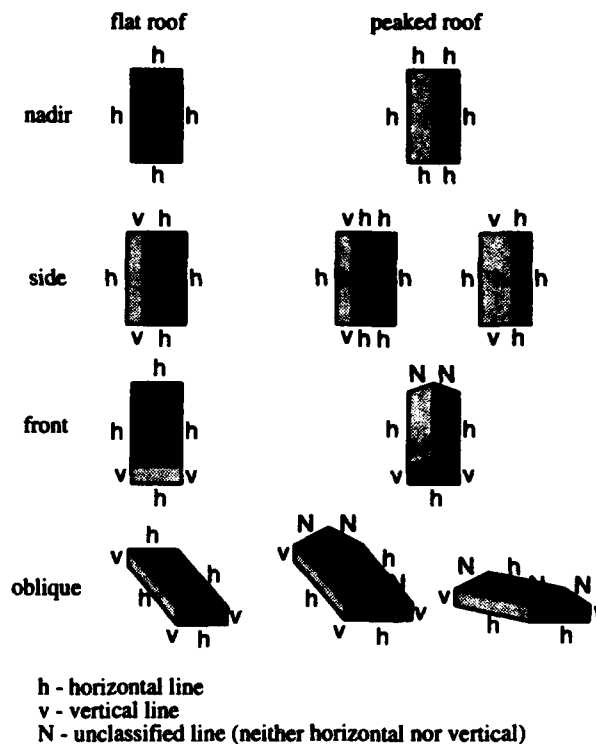


Figure 5: Simple building model.

problem by generating building hypotheses, i.e., boxes, for every subchain of edges in a chain. This is accomplished by taking every subchain of at least two edges and completing them to four-sided boxes.

Typically, only about 10% of the boxes generated for a scene correspond to buildings. BABE's verification phase selects building candidates from the boxes generated in the previous phase. It performs this task by examining the boxes for indications of a shadow region along the shadow casting edges.

Under an oblique viewing geometry, BABE's model first breaks down in the corner detection phase where right-angled corners in the scene may not translate to right-angled corners in the image. In fact, the actual angle depends not only on the obliquity of the viewing geometry, but on the relative position and orientation of the building in the scene.

Using the horizontal and vertical line identification techniques described in Section 3, we can assign attributions to each edge prior to corner generation. We can then make use of a simple building model, outlined in Figure 5. This model presents two simple and common classes of buildings, those



Figure 6: BABE hypotheses, RADT9WOB.

with flat roofs and those with peaked roofs. The two types of buildings are shown from various viewpoints (symmetric cases are omitted for brevity).

Each distinct line segment in the diagram has been assigned a label, indicating whether it is a vertical or horizontal line in object space, or whether it is neither. In object space, we observe that for flat-roof structures, side and front facets of buildings are instances of rectangles composed of alternating horizontal and vertical segments, and roof facets are instances of rectangles formed by four horizontal segments. For peaked-roof structures, each side facet is again represented by a rectangle of alternating horizontal and vertical segments; roof facets are now instances of rectangles of alternating horizontal and unlabeled segments. A front facet of a peaked-roof structure is a pentagon, composed of two unlabeled segments, two verticals, and a horizontal segment.

It is worth noting that BABE does not explicitly use this simple model in its processing phases; there is nothing in principle that prohibits an extension to BABE for constructing more complex shapes by joining these rectangular or pentagonal facets. The model is useful, however, for visualizing the relationships between horizontal, vertical, and unlabeled lines in typical man-made structures.

These properties of building facets suggest the following set of heuristics for corner detection:



Figure 7: Geometrically consistent hypotheses.

- Two intersecting verticals never form a valid corner in object space.
- A horizontal-vertical intersection is allowed to form a corner.
- Two intersecting horizontals are allowed to form a corner, if their intersection in object space forms a right angle.
- An unlabeled line intersecting with a labeled line is allowed as a corner, since it is potentially part of a peaked roof.
- Two intersecting unlabeled lines are allowed to form a corner, as they may be part of a pentagonal facet; it should be noted, however, that the current version of BABE will not generate pentagonal descriptions. We intend to pursue more general shape constructions in future work.

These heuristics must take into account the fact that a given line may be labeled as both horizontal and vertical, if the imaging geometry is such that the direction of the horizontal vanishing point for some set of lines is the same as the vertical vanishing point. They do so by allowing such lines to be regarded as both horizontal and vertical lines during corner formation.

6. Building hypothesis generation

Given the ability to generate corners in oblique imagery, BABE can be used to generate structural hypotheses, boxes which delineate structure in the scene. In the original implementation of BABE, the only geometric constraint applied during line-

corner linking and box formation was the right-angle constraint on corners. In the new implementation, we can apply our simple building model at this stage to prune geometrically inconsistent hypotheses.

For each box generated by BABE, we examine the horizontal and vertical line attributions assigned to each line segment of the box. If the four attributions are consistent with the labelings of any building facet in the building model, the box is accepted. One example would be a facet with alternating horizontal and vertical lines, which is consistent with a side facet of a building. If the four attributions do not match any of the allowable building facets, the box is rejected as being geometrically inconsistent. One example of this situation would be a box comprised of four vertical lines; such a facet is impossible in the model.

Figure 6 shows the complete set of boxes generated by BABE prior to the application of geometric labeling constraints; in this case, there are 2899 boxes. Figure 7 shows the set of 628 boxes left after the labeling constraints have been exercised. As the figures show, the labeling constraints alone provide a strong constraint on the permissible hypothesis geometries.

After the application of the labeling constraints, the boxes are passed through BABE's verification phase, which estimates shadow intensity and sun illumination direction and uses this knowledge to score each hypothesis based on its conformance with these parameters. At this time, the verification phase makes no use of the photogrammetric information, and hence treats all hypotheses as though they represented features in a nadir-acquisition geometry. We intend to address this shortcoming in future work.

At this stage, we are left with a set of hypotheses which are presumed to be geometrically consistent, in that they are composed of corners exhibiting valid angles in image space and that they possess valid labelings with respect to our simple building model, and which are presumed to be photometrically consistent, in that they exhibit a combination of strong intensity gradient across edge boundaries and are adjacent to dark regions in the image which could plausibly be the shadows of the hypothesized structures.

Given these presumptions, it is reasonable to regard these hypotheses as verified facets of three-dimensional structure in the scene. Using the scene geometry in conjunction with our building model, it becomes possible to extrapolate these partial delineations of building structure into more complete building models. We consider one such extrapolation here, that of completing partially peaked roofs to cover the entire roof. Using our model, we know that facets with alternating unlabeled and horizontal lines must be peaked roof facets; we can detect these facets by examining the line labelings and applying geometric constraints to extrapolate the other peaked roof facet in the pair.

Figure 8 illustrates the situation at hand. The hypothesized facet represents a BABE hypothesis which we wish to use as a guide for hypothesizing the other half of the rooftop. We begin by computing the line perpendicular to the horizontal line R in object space, and projecting this perpendicular into image space (line C). Next, we intersect that line with the line drawn through the roof peak point p and the vertical vanishing point vp , to obtain a point x . In object space, the distance between x and e is equal to the distance between x and n ; we assume that these distances are equal in image space as well, and complete the new building facet by using the roof peak point p , points n and f , and the application of symmetry to generate g .

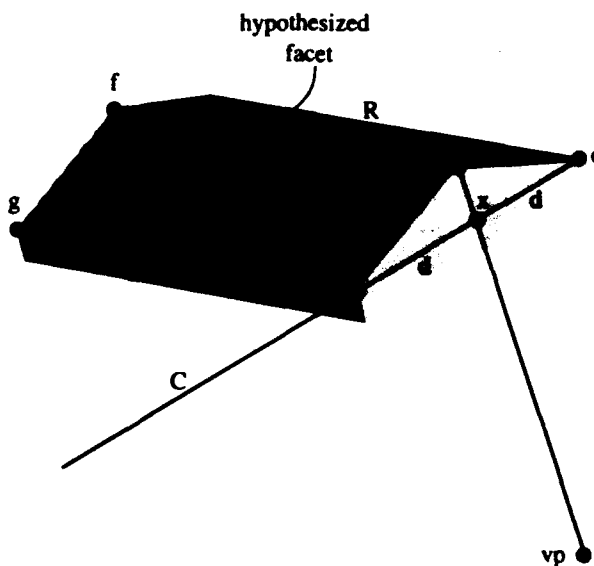


Figure 8: Peaked roof projection.



Figure 9: Original BABE results.

Figure 9 shows the original BABE results for the scene; Figure 10 illustrates the final result generated by our current extensions to BABE. Two improvements are apparent from the figures; the geometric consistency pruning has eliminated many spurious boxes generated by accidental alignments of small edge features, and the peak projection technique has improved the modeling of peaked structures in this scene, correctly hypothesizing roof facets that were either lost in the shadow evaluation phase of BABE, or were never generated due to a lack of edge information. There are still problems; one of the false hypotheses on the right side of Figure 10 has a long narrow facet, generated by the peak projection technique. The immediate problem is that the supposed horizontal roof edge line is perpendicular to the vertical lines in image space, and hence the line through the roof peak and the vanishing point will be nearly parallel to the roof edge line, resulting in a facet hypothesis whose roof edge will be very far from the original facet.

The problem just described arises in part from the fact that the facet projection technique is only an approximation to the true image space relationships between edges, and this approximation breaks down when structures possess horizontal roof lines and vertical lines with the same orientation in image space. Ultimately, however, the problem is due to issues in modeling and hypothesis generation. In a full

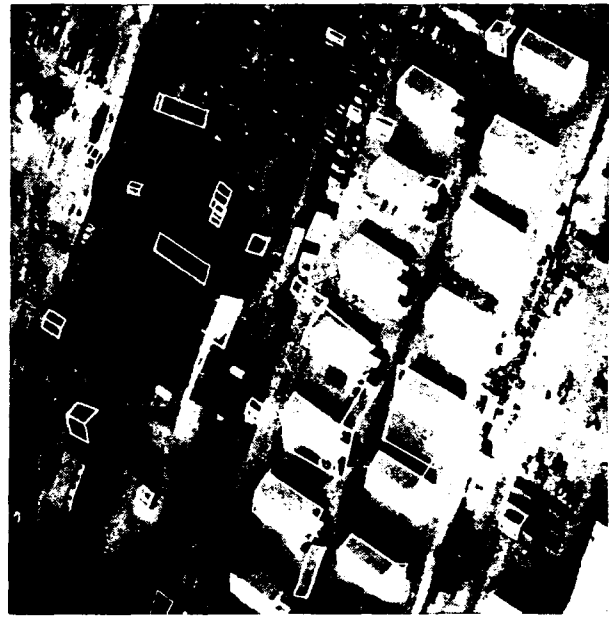


Figure 10: New BABE results.

implementation of a general viewpoint BABE, it would be desirable to maintain the generate-and-test paradigm used in the original version of BABE. During the line-corner chain forming phase, one would like to construct full three-dimensional structural models in object space, rather than two-dimensional models in image space. These models would then be subjected to a verification process similar in spirit to the shadow constraint algorithms BABE now employs, but with the added information provided by scene geometry and illumination constraints on adjacent planar surfaces. This point will be discussed again in the final section.

7. Analysis and future work

Preliminary results from the inclusion of geometric and metric knowledge into the building extraction system have been promising, although they have highlighted the limitations of the current implicit building models within the BABE system. We believe that these limitations are typical of other building extraction research based upon nadir view assumptions.

Our experimentation has been limited to five test areas visible in each of four images of Fort Hood. Two of the images have near-nadir geometry, while two are oblique with a relatively wide angle field of view. We expect to continue to refine and validate our research on a wider set of imagery. Some specific observations regarding our work are as follows:



Figure 11: Multiple view verification.

- The height estimates for the candidate vertical lines are good refinement and information fusion cues, since the object-space measurements can be directly compared with other sources of heights, such as shadows. A next step is to incorporate precision information on the measurements into an information fusion framework [Shufelt 93] to allow for relative weighting of the measurements.
- The height and length attributions can also be used to constrain hypothesis search by pruning unrealistic building sizes.
- A small catalog of structural formation constraints (Figure 5) can be a powerful tool for pruning hypotheses.
- Verification of horizontal and vertical lines across multiple views can reduce the number of hypotheses for later processing stages, thereby increasing efficiency. A future step is to perform multiple view verification of corners, which should also decrease the number of hypotheses and improve their quality. Figure 11 shows the BABE results using horizontal edges verified with another image. From 1634 original horizontal edges, 478 were eliminated because of length (less than 10 meters) and 141 were eliminated because they did not match another horizontal edge on the overlapping image. A comparison of Figure 11 with Figure 10 shows 30 fewer building hypotheses generated, with only one of the structures eliminated actually corresponding to a real structure in the scene.
- Although BABE has been designed as a monoscopic system, the capability of precisely

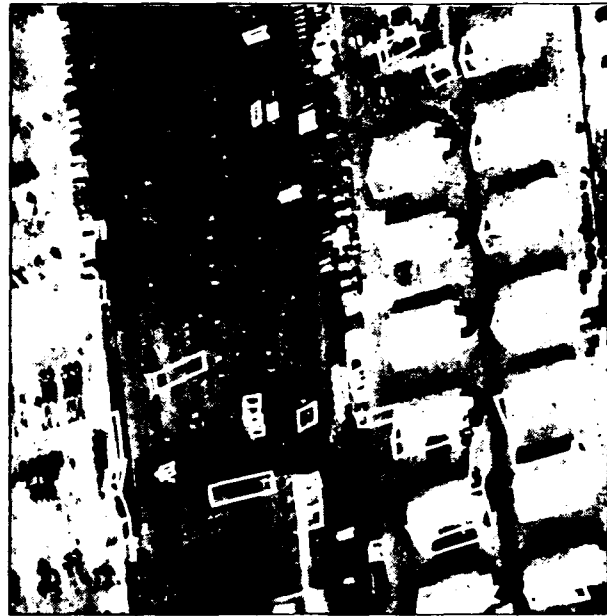


Figure 12: Reprojected BABE results.

combining multiple views using the photogrammetric information allows the hypothesis generation and verification to take place completely in object space. Figure 12 shows the BABE results for test area RADT9WOB projected into another image, using the camera resection results for the two images. This illustrates the advantages of being able to tie images together with rigorous camera models, especially required for oblique imagery.

- The BABE model can also be extended to handle illumination constraints on the building facets (such as variation across peaked roofs of uniform material, given the sun angle), more rigorous shadow detection and verification, and stereo disparity.

Results thus far have shown the need for true three-dimensional modeling of object structure. Figures 13 and 14 compare the original and new BABE hypotheses for another of our test scenes. In this case, the new data are qualitatively much worse, despite the system's ability in a few instances to correctly extrapolate roof structure. The problem here arises in BABE's hypothesis verification algorithms. In the old version of BABE, no geometric consistency checks are performed on the set of boxes before they are passed to the verification phase. In the new version, geometric pruning presents the verification phase with a smaller set of boxes for statistical analysis, and the verification algorithms choose a higher cutoff for plausible hypotheses.



Figure 13: Original BABE results, RADTSWOB.

eliminating many true boxes. This can in part be blamed on the adaptive nature of the verification scoring algorithms in BABE, but the problem is symptomatic of the larger issue of modeling. Ideally, the generation and verification algorithms would work with three-dimensional models in object space. This strategy allows all feasible models to reach verification, where precise geometric information permits rigorous testing of illumination constraints across adjacent planar surfaces; prediction and verification of cast shadows [Irvin 89]; and, the application of stereoscopic information for consistency constraints across multiple views. Understanding these issues and the evaluation of rigorous techniques to address these problems are the subject of current research.

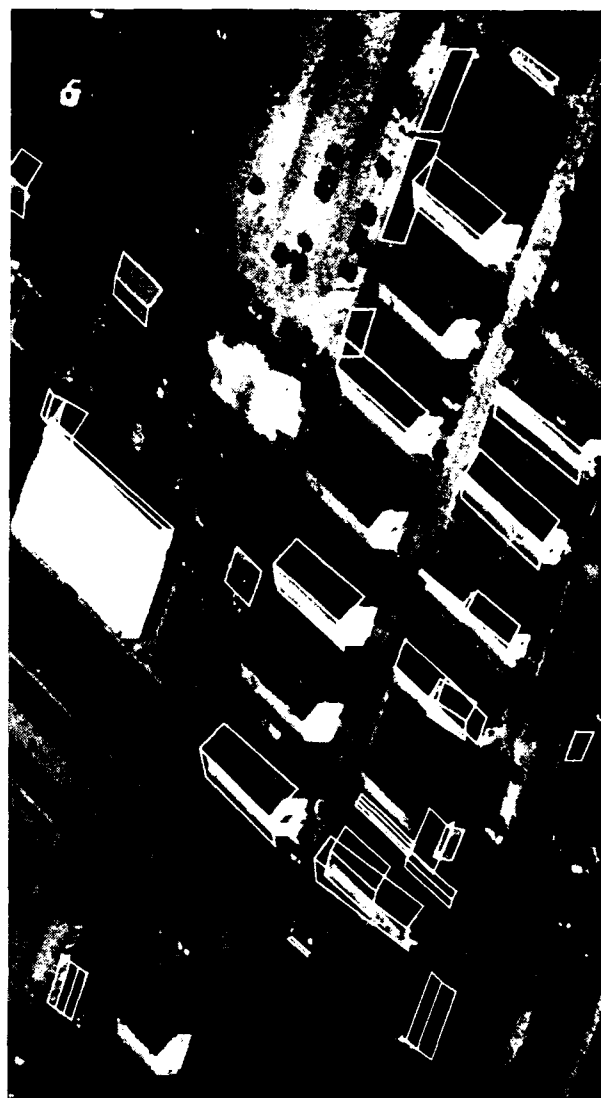


Figure 14: New BABE results, RADTSWOB.

8. Conclusion

We have described initial experiments in incorporating photogrammetric calculations in an existing building detection system, analyzed the results on a small set of oblique aerial images, and raised several issues of modeling, hypothesis generation, and hypothesis verification that must ultimately be addressed in a complete implementation of a photogrammetrically rigorous feature extractor. Although the work is certainly in its preliminary stages, and many issues and flaws remain to be addressed, we believe that the qualitative results show that the combination of precise camera modeling and geometric information with existing feature extraction algorithms provides a powerful approach for increasing the performance of building detectors on complex aerial imagery.

Acknowledgments

We would like to acknowledge the efforts of Steven Cochran, Stephen Lacy, and Mark Stemmm in measuring the image coordinates for the resection, DARPA for providing challenging imagery of an area we always wanted to visit, and the Pizza Outlet for saving the authors from cannibalism during the final stretch.

References

- [Barnard 83]
Stephen T. Barnard.
Interpreting perspective images.
Artificial Intelligence 21:435-462, 1983.
- [Brillault 92]
B. Brillault-O'Mahony.
High level 3D structures from a single view.
Image and Vision Computing 10(7):508-520, September, 1992.
- [Huertas 88]
A. Huertas and R. Nevatia.
Detecting buildings in aerial images.
Computer Vision, Graphics, and Image Processing 41:131-152, April, 1988.
- [Irvin 89]
R. B. Irvin and D. M. McKeown.
Methods for exploiting the relationship between buildings and their shadows in aerial imagery.
IEEE Transactions on Systems, Man and Cybernetics 19(6):1564-1575, November, 1989.
- [Lebegue 92]
Xavier Lebegue and J. K. Aggarwal.
Detecting 3-D parallel lines for perceptual organization.
In *Proceedings of the European Conference on Computer Vision*, pages 720-724. 1992.
- [Liow 90]
Y.-T. Liow and T. Pavlidis.
Use of shadows for extracting buildings in aerial images.
Computer Vision, Graphics, and Image Processing 49:242-277, 1990.
- [McKeown 90]
D. M. McKeown.
Toward automatic cartographic feature extraction.
In L. F. Pau (editor), *NATO ASI Series. Volume F 65: Mapping and Spatial Modelling for Navigation*, pages 149-180. Springer-Verlag, Berlin Heidelberg, 1990.
- [Mohan 89]
R. Mohan and R. Nevatia.
Using perceptual organization to extract 3-d structures.
IEEE Transactions of Pattern Analysis and Machine Intelligence 11(11):1121-1139, November, 1989.
- [Nevatia 80]
R. Nevatia and K. R. Babu.
Linear feature extraction and description.
Computer Graphics and Image Processing vol. 13:257-269, July, 1980.
- [Nicolin 87]
B. Nicolin and R. Gabler.
A knowledge-based system for the analysis of aerial images.
IEEE Transactions on Geoscience and Remote Sensing GE-25(3):317-329, May, 1987.
- [Shufelt 93]
Jefferey A. Shufelt and David M. McKeown.
Fusion of monocular cues to detect man-made structures in aerial imagery.
Computer Vision, Graphics, and Image Processing: Image Understanding 57(2), March, 1993.
- [Slama 80]
Chester C. Slama (editor).
Manual of Photogrammetry.
American Society of Photogrammetry, 1980.

Measuring the Affine Transform - I: Recovering Scale and Rotation

R. Manmatha and John Oliensis *

Computer Science Department

University of Massachusetts at Amherst, Amherst, MA-01003

manmatha@cs.umass.edu

Abstract

Image deformations due to relative motion between an observer and an object may be used to infer 3-D structure. Up to first order these deformations can be written in terms of an affine transform. This paper presents a method for measuring the affine transform locally between two image patches that correctly handles the difficulty of finding the correspondence between deformed patches. The method uses weighted moments of brightness. It is shown that the moments of deformed patches are related through functions of affine transforms. In the special case where the affine transform can be written as a scale change and an in-plane rotation, the zeroth and first moment equations are solved for the scale. The robustness of the method is demonstrated experimentally.

1 Introduction

Changes in the relative orientation of a surface with respect to a camera cause deformations in the image of the surface. Deformations can be used to infer local surface geometry from motion [Koenderink and van Doorn, 1987; Sawhney and Hanson, 1991; Cipolla and Blake, 1992; Jones and Malik, 1992b]. Since a repeating texture pattern can be thought of as a pattern in motion, shape from texture can also be derived from deformations [Kanade and Kender, 1983; Super and Bovik, 1992]. Constraints on the shape of the undeformed structure also allow the computation of shape from texture [Brown and Shyvester, 1990; Garding, 1990].

To first order, the image deformation and translation due to relative motion can be described using a six parameter affine transformation A where

$$Ar = \begin{vmatrix} u_0 \\ v_0 \end{vmatrix} + \begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix} \begin{vmatrix} r_x \\ r_y \end{vmatrix}. \quad (1)$$

*This research was supported in part by DARPA (via TACOM) under contract number DAAE07-91-C-R035, and by the National Science Foundation under grants CDA-8922572 and IRL-9113690.

r_x and r_y are the image coordinates and u_0 and v_0 the image translation. This is a valid approximation assuming local planarity and weak perspective projection [Kanade and Kender, 1983]. Even in situations where full-perspective projection must be used, it can be shown that if the change in relative orientation of the surface patches is small, the image projections can again be related by an affine transform [Adiv, 1985].

The recovery of 3-D structure from the affine transform requires robust *local* estimates of the affine parameters. Consider the case where an image patch F_1 is deformed into a patch F_2 (either in the same image or in another image) by an unknown affine transform. The problem of measuring the affine transform is to first find the corresponding patch F_2 given F_1 and second to recover the affine parameters from the two patches. Even if the centroids of the image patches are matched, the precise size and shape of F_2 is difficult to determine since it is a function of the unknown deformation. If this correspondence is not precisely done, the affine parameters will be determined incorrectly. Thus the problem is more difficult than in standard correspondence problems e.g. the determination of optical flow.

Existing methods using image patches usually ignore this problem. A number of techniques assume that the affine parameters are small and then linearise the brightness function or filtered versions of it with respect to the spatial coordinates [Bergen *et al.*, 1992; Koenderink and van Doorn, 1987; Campani and Verri, 1992; Werkhoven and Koenderink, 1990]. Thus these methods are restricted to cases where the affine transform is small which in turn requires that the 3-D motion be small. [Jones and Malik, 1992b] do not assume that the affine transform is small. Their method, however, uses brute force search techniques and again ignores the determination of precise correspondence. A natural way to find correspondence is to use straight lines [Sawhney and Hanson, 1991] or closed boundary contours [Cipolla and Blake, 1992], with the change in the size and shape of the enclosed area defining the affine transform. These methods, however, fail when such structures are absent as in many richly textured

scenes. Further, their use has only been demonstrated on homogeneous image regions with closed boundaries.

This paper presents a technique for reliably measuring affine transforms that correctly handles the difficulty of corresponding deformed image patches. The image patches are filtered using gaussians and derivatives of gaussians. Measuring the affine transform is then recast as a problem of finding the deformation parameters of the filters rather than the patches. For example, let F_1 and F_2 be related by a scale change s . Then the output of F_1 filtered with a gaussian of σ will be equal to the output of F_2 filtered with a gaussian of $s\sigma$. Similar relationships hold for arbitrary affine transforms and filters described by derivatives of gaussians. These equations are exact for any arbitrary affine transform in arbitrary dimensions.

The second part of the paper focuses on solving for the affine transform when it can be written as the product of a scale change and a rotation (the solution for the general case will be considered in future papers). For example, this situation arises in the case of mostly translational camera motion and shallow structures (i.e. structures whose extent in depth is small compared to their distance from the camera [Sawhney and Hanson, 1991]).

The equation can be solved by sampling the σ space. Rather than use a brute force search technique, the search space is sampled for a few different σ' and one of the σ' is picked as the operating point. The scale is recovered by linearising the gaussian filter with respect to σ about this operating point using the diffusion equation. Consistency is used to establish the correct operating point. Note that *linearization is done with respect to σ as opposed to linearisation with respect to the image coordinates* done by other methods. As a result, scale changes of arbitrary magnitude can be dealt with by choosing different operating points. The rotations can also be arbitrary. In contrast, linearising with respect to the image coordinates is a valid approximation only for small affine transforms.

The gaussian (zereth moment) equation is linear in the scale parameter. By sampling at several scales, an overconstrained linear system is obtained. This is solved for scale using singular value decomposition. Using the first moment an equation which is nonlinear with respect to scale is obtained. Again, this may be sampled at multiple scales to provide an overconstrained system of equations. This non-linear system is solved using the Gauss-Newton technique. The first moment equation also allows the computation of the rotation. Both the formulation and the solution are done for arbitrary dimensions, not just 2. Experimental results are shown on both synthetic and real images attesting to the robustness and simplicity of the method.

2 Deformation of Filters: Zero Moments

Notation Vectors will be represented by lowercase letters in boldface while matrices will be represented by uppercase letters in boldface.

We will assume that the image translation is known and has been set to zero. Methods for finding the image translation are briefly discussed in section 4.2.4. Then the affine transform has only four deformation parameters. It is also assumed that shading and illumination effects can be ignored. These can, however, be taken care of by incorporating an additional constant factor in the equations. For simplicity, we focus on the 2-D case although the discussion is dimension-independent.

Our discussion is based on two observations. First, the result of a filtering operation on two image patches will be different in general unless the filter is appropriately deformed for the second image patch—the deformation being a function of the affine transform. Second, moments of the image patches are related by simple functions of the affine transforms, and this can be exploited to compute the affine transform.

Consider two functions F_1 and F_2 related by an affine transform of the underlying coordinate system. Then

$$F_1(\mathbf{r}) = F_2(\mathbf{A}\mathbf{r}) \quad (2)$$

Their integrals over some finite interval are related by:

$$\begin{aligned} \int_{-\mathbf{a}}^{\mathbf{a}} F_1(\mathbf{r}) d\mathbf{r} &= \int_{-\mathbf{a}}^{\mathbf{a}} F_2(\mathbf{A}\mathbf{r}) d\mathbf{r} \\ &= \int_{-\mathbf{A}\mathbf{a}}^{\mathbf{A}\mathbf{a}} F_2(\mathbf{A}\mathbf{r}) d(\mathbf{A}\mathbf{r}) \det \mathbf{A}^{-1} \quad (3) \end{aligned}$$

expressed succinctly as

$$\nu_1 = \nu_2 \times \det \mathbf{A}^{-1}. \quad (4)$$

Let s_i be the scale change along the i^{th} dimension and n the number of dimensions. Then $\det(\mathbf{A}) = \prod_1^n s_i$. This can be intuitively understood as follows. Consider the 1-D case ($n = 1$), where the affine transform reduces to a scale change. Let the function F_1 be graphed on a rubber sheet. The graph of F_2 is obtained by stretching the sheet and attached coordinate system. The determinant term is equal to the stretching undergone by the coordinates. Note that the integral of a function may also be viewed as its zeroth moment.

(3) cannot be used directly because the limits on the right-hand side depend on the affine transform and are therefore unknown. This crucial point has not been handled correctly before. On the other hand, taking the limits from $-\infty$ to ∞ would not preserve localisation. The solution to this problem is to weight the function by another which decays rapidly—here the gaussian is used.

We present the weighted equations analogous to (3) first for the case where $A = sR$ (i.e. the affine transform equals a scale change s times a rotation R), followed by the general case.

2.1 Case $A = sR$

Denote the unnormalised gaussian by

$$H(r, \sigma^2) = \exp(-r^T r / 2\sigma^2). \quad (5)$$

Multiply both sides of (2) by $H(r, \sigma^2)$ to obtain

$$F_1(r)H(r, \sigma^2) = F_2(Ar)H(r, \sigma^2). \quad (6)$$

From the orthonormality of rotations it follows that

$$H(r, \sigma^2) = H(Ar, (s\sigma)^2), \quad (7)$$

which allows (6) to be rewritten as

$$F_1(r)H(r, \sigma^2) = F_2(Ar)H(Ar, (s\sigma)^2). \quad (8)$$

The weighted zeroth moment is therefore

$$\begin{aligned} \int F_1(r)H(r, \sigma^2)dr &= \int F_2(Ar)H(Ar, (s\sigma)^2)dr \\ &= \int F_2(Ar)H(Ar, (s\sigma)^2)d(sRr)s^{-n}, \end{aligned} \quad (9)$$

where the limits are taken from $-\infty$ to ∞ . The factor $s^{-n} = \det A^{-1}$ can be eliminated by using normalised gaussians

$$G(r, \sigma^2) = \frac{1}{(2\pi)^{n/2}\sigma^n} H(r, \sigma^2) \quad (10)$$

in place of H . The moment equation then becomes

$$\begin{aligned} \int F_1(r)G(r, \sigma^2)dr &= \frac{1}{(2\pi)^{n/2}\sigma^n} \int F_2(Ar)H(Ar, (s\sigma)^2)d(Ar)s^{-n} \\ &= \int F_2(Ar)G(Ar, (s\sigma)^2)d(Ar). \end{aligned} \quad (11)$$

The integral may be interpreted as a gaussian convolution or filtering at the origin. Thus we write (11) as

$$F_1 * G(r, \sigma^2) = F_2 * G(r_1, (s\sigma)^2), \quad (12)$$

where $r_1 = Ar$.

(12), the weighted analog of (3), is exact and valid for arbitrary dimensions. The problem of recovering the affine parameters has been reduced to finding the deformation of a known function, the gaussian, rather than that of the unknown brightness functions. However since (12) is invariant to rotation it can only be used for recovering the scale (the recovery of rotation by other means is discussed later). Note that although the limits are infinite, since the gaussian is a rapidly decaying function, it suffices in practice to take limits from -4σ to 4σ (and correspondingly from $-4s\sigma$ to $4s\sigma$ on the right-hand side).

2.2 General Case

A similar equation can be shown to hold for arbitrary affine transforms, provided generalised gaussians are used. Define a generalised gaussian as

$$G(r, M) = \frac{1}{(2\pi)^{n/2}\det(M)^{1/2}} \exp\left(-\frac{r^T M^{-1} r}{2}\right) \quad (13)$$

where M is a symmetric positive semi-definite matrix. Then

$$\begin{aligned} G(r, \sigma^2 I) &= \frac{1}{(2\pi)^{n/2}\det(\sigma^2 I)^{1/2}} \exp\left(\frac{-r^T r}{2\sigma^2}\right) \\ &= \frac{1}{(2\pi)^{n/2}\sigma^n} \exp\left(-\frac{(Ar)^T (AA^T)^{-1} (Ar)}{2\sigma^2}\right) \\ &= \det(AA^T)^{1/2} G(Ar, \sigma^2 (AA^T)). \end{aligned} \quad (14)$$

Thus the weighted moment equation may be written

$$\begin{aligned} \int F_1(r)G(r, \sigma^2 I)dr &= \det(A) \times \\ &\int F_2(Ar)G(Ar, \sigma^2 (AA^T))d(Ar)\det(A^{-1}) \\ &= \int F_2(Ar)G(Ar, \sigma^2 (AA^T))d(Ar), \end{aligned} \quad (15)$$

where the identity $\det(AA^T)^{1/2} = \det(A)$ has been used. The matrix AA^T is a symmetric, positive semi-definite matrix and may therefore be written

$$\frac{1}{\sigma^2} AA^T = RER^T, \quad (16)$$

where R is a rotation matrix and E a diagonal matrix with entries $s_1\sigma^2, s_2\sigma^2, \dots, s_n\sigma^2$ ($s_i \geq 0$). Thus

$$\int F_1(r)G(r, \sigma^2 I)dr = \int F_2(Ar)G(Ar, RER^T)d(Ar) \quad (17)$$

Again, to show the connections to convolution and filtering, this may be written as

$$F_1 * G(r, \sigma^2 I) = F_2 * G(r_1, RER^T). \quad (18)$$

(18) is the analog of the zero moment equation (3), and can be used for determining the affine transform. The level contours of the generalised gaussian are ellipsoids rather than spheres. The tilt of the ellipsoid is given by the rotation matrix while its eccentricity is given by the matrix E , which is actually a function of the scales along each dimension. (18) clearly shows that to recover affine transforms by filtering, one must deform the filter appropriately; a point ignored in previous work [Bergen et al., 1992; Koenderink and van Doorn, 1987; Campani and Verri, 1992; Werkhoven and Koenderink, 1990; Jones and Malik, 1992b]. The zero moment equation (18) alone does not permit the recovery of the complete affine matrix—only the scales and the tilt. To find the complete affine transform, higher order moments need to be considered. Using higher order moments also permits the use of more overconstrained equations.

3 Higher Order Moments

The first order moments of F_1 and F_2 are related by

$$\begin{aligned} \int_{-\mathbf{a}}^{\mathbf{a}} \mathbf{r} F_1(\mathbf{r}) d\mathbf{r} &= \int_{-\mathbf{a}}^{\mathbf{a}} \mathbf{r} F_2(\mathbf{A}\mathbf{r}) d\mathbf{r} \\ &= \mathbf{A}^{-1} \int_{-\mathbf{A}\mathbf{a}}^{\mathbf{A}\mathbf{a}} \mathbf{A}\mathbf{r} F_2(\mathbf{A}\mathbf{r}) d(\mathbf{A}\mathbf{r}) \det \mathbf{A}^{-1} \end{aligned} \quad (19)$$

or

$$\mu_1 = \mathbf{A}^{-1} \mu_2 \times \det \mathbf{A}^{-1}. \quad (20)$$

The second order moments are given by

$$\begin{aligned} \int_{-\mathbf{a}}^{\mathbf{a}} \mathbf{r} \mathbf{r}^T F_1(\mathbf{r}) d\mathbf{r} \\ = \mathbf{A}^{-1} \int_{-\mathbf{A}\mathbf{a}}^{\mathbf{A}\mathbf{a}} \mathbf{A}\mathbf{r} (\mathbf{A}\mathbf{r})^T F(\mathbf{A}\mathbf{r}) d(\mathbf{A}\mathbf{r}) (\mathbf{A}^{-1})^T \det \mathbf{A}^{-1} \end{aligned}$$

and this may be expressed as

$$\Gamma_1 = \mathbf{A}^{-1} \Gamma_2 (\mathbf{A}^{-1})^T \det(\mathbf{A}^{-1}). \quad (21)$$

Note that the seroth moment equation is a scalar equation, (20) a vector one, and (21) is a matrix equation. As before, the moment equations (20) and (21) are not directly usable due to the difficulty that the limits of the patches integrated over depend on the deformation. Therefore we again employ gaussian weighted moments, using the fact that the derivatives of gaussians are closely related to moments weighted with gaussians.

The effect of filtering with derivatives of gaussians can be obtained by differentiating the gaussian (13). First write $\mathbf{r}_1 = \mathbf{A}\mathbf{r}$. Differentiating (13) gives

$$F_1(\mathbf{r}) * dG(\mathbf{r}, \sigma^2 \mathbf{I}) / d\mathbf{r} = \mathbf{A}^T F_2(\mathbf{r}_1) * dG(\mathbf{r}_1, \mathbf{A} \mathbf{A}^T \sigma^2) / d\mathbf{r}_1 \quad (22)$$

where

$$dG(\mathbf{r}, \sigma^2 \mathbf{I}) / d\mathbf{r} = -\frac{\mathbf{r}}{\sigma^2} G(\mathbf{r}, \sigma^2 \mathbf{I}) \quad (23)$$

and

$$dG(\mathbf{r}_1, \mathbf{A} \mathbf{A}^T \sigma^2) / d\mathbf{r}_1 = -(\mathbf{A} \mathbf{A}^T \sigma^2)^{-1} \mathbf{r}_1 G(\mathbf{r}_1, \mathbf{A} \mathbf{A}^T \sigma^2). \quad (24)$$

This equation looks different from the first moment (20) because the first derivative of the gaussian has been normalised. Convolving with second derivatives of a gaussian gives

$$\begin{aligned} F_1(\mathbf{r}) * d^2 G(\mathbf{r}, \sigma^2 \mathbf{I}) / d(\mathbf{r} \mathbf{r}^T) = \\ \mathbf{A}^T F_2(\mathbf{r}_1) * d^2 G(\mathbf{r}_1, \mathbf{A} \mathbf{A}^T \sigma^2) / d(\mathbf{r}_1 \mathbf{r}_1^T) \mathbf{A} \end{aligned} \quad (25)$$

where

$$d^2 G(\mathbf{r}, \sigma^2 \mathbf{I}) / d(\mathbf{r} \mathbf{r}^T) = \frac{(\mathbf{r} \mathbf{r}^T) / \sigma^2 - \mathbf{I}}{\sigma^2} G(\mathbf{r}, \sigma^2 \mathbf{I}) \quad (26)$$

and

$$\begin{aligned} d^2 G(\mathbf{r}_1, \mathbf{A} \mathbf{A}^T \sigma^2) / d(\mathbf{r}_1 \mathbf{r}_1^T) = \\ [(\mathbf{A}^T \mathbf{A} \sigma^2)^{-1} \mathbf{r}_1 \mathbf{r}_1^T (\mathbf{A}^T \mathbf{A} \sigma^2)^{-1} - (\mathbf{A} \mathbf{A}^T)^{-1}] \times \\ G(\mathbf{r}_1, \mathbf{A} \mathbf{A}^T \sigma^2). \end{aligned} \quad (27)$$

(25) and (21) are seen to be closely related; the differences are the additional term due to the gaussian in (25) and due to normalisation.

Since convolutions with gaussians and derivatives of gaussians are so closely related to the original weighted moment equations, they will often be referred to as moments in the rest of the paper.

4 Solving the Moment Equations

4.1 The problem of even and odd functions F

If the value of the moments is zero, (or near zero in practice), the moment equations are ill-conditioned and cannot be solved. This can occur in two ways; either the signal strength is too low (i.e. the magnitude of F is small) or the function F is purely even or purely odd causing some of the moment equations to be zero. There is little that can be done in the first case. The latter case, however, provides insight into the number of moment equations required to solve for the affine parameters.

Consider first the 1-D case. It is easy to see that the even moments of any odd function will be zero while the odd moments of any even function are zero. Since the seroth moment is even, and the first moment is odd and only one parameter (the scale) needs to be determined, these two equations suffice to find it.

The situation is a little more complicated in higher dimensions. One way of stating the problem is to consider each dimension separately. Then if a function is odd along any dimension, its contribution to the even moment from that dimension will be zero and hence inferences along that dimension cannot be made. Similarly, if a function is even along any dimension, its contribution is zero to the odd moments along that dimension. Note that typically a function is even or odd only at a few points over its domain, so this may not be a significant problem.

How many moments are required in 2-D to solve for the affine transform? In general four affine parameters need to be determined. Straightforward equation counting seems to show that there are four even equations (1 from the seroth moment and 3 from the second moment), and there are six odd equations (2 from the first moment and 4 if the third moment is used). Thus even if the function is purely even or odd, moments up to third order suffice to solve for the affine parameters.

However, the third moment is actually not required, since the previous analysis ignores the information

available from the deformation of a gaussian. Consider the seroth moment when an arbitrary affine transform A needs to be measured. In this case, the seroth moment may be used to find the matrix AA^T (i.e. 3 parameters may be computed). Accounting for the additional information available from the deformation of the gaussian, there are at least 6 even equations (3 from the seroth moment and at least 3 from the second moment) and at least 4 odd equations (from the first moment).

The function F may also be transformed so that some of the moments are always non-zero. For example, if instead of the function F , the magnitude of its auto-correlation is used, the seroth and the second moment are always non-zero. This follows from the radial symmetry of the auto-correlation function—which implies that the odd moments are all zero while the even moments are non-zero.

A different transformation uses certain algebraic tricks to convert any function to an odd or an even function thus ensuring that every moment equation is well-defined. For example, consider the 1-D case again. Every function F can be written as the sum of an even part EF and an odd part OF . The odd part OF can be converted into an odd function by taking its magnitude. The even part EF can be converted to an odd function by flipping one half of the function. The problem is somewhat more complicated in higher dimensions. The 2-D case will be dealt with in the solution section.

4.2 Solving the Zeroth Moment Equation

In the remainder of this paper, only the case where the affine transform $A = sR$ (i.e. a scale change and a rotation) will be considered (see section 2.1); the general affine transform will be considered in a later paper. The seroth moment equation will be dealt with first followed by the first moment equation.

When the affine transform is described by $A = sR$, (12) can be written as

$$F_1(r) * G(r, \sigma^2) = F_2(r_1) * G(r_1, (\sigma^*)^2) \quad (28)$$

where $\sigma^* = s\sigma$. The important point here is that the rotation matrix does not figure in σ^* . The problem of finding the scale parameter has therefore been converted into the problem of finding the value of σ^* . Older methods [Bergen et al., 1992; Koenderink and van Doorn, 1987; Campani and Verri, 1992; Werkhoven and Koenderinck, 1990; Jones and Malik, 1992b] have instead concentrated on the much more difficult problem of trying to correspond the functions F_1 and F_2 .

The equation can be solved by sampling the space of possible values of σ^* , filtering for each sampled value and declaring the solution to be that value of σ^* for

which the above equation has smallest residual error according to some norm. A more elegant approach uses the fact that the affine transform can be analytically interpolated. The idea is to sample over a small set of σ^* and then interpolate using a Taylor series approximation. Consider first a given σ^* . The Taylor series approximation to first order gives

$$\begin{aligned} G(r_1, (s\sigma)^2) &\approx G(r_1, \sigma^2) + \alpha\sigma \frac{\partial G(r_1, \sigma^2)}{\partial \sigma} \quad (29) \\ &= G(r_1, \sigma^2) + \alpha\sigma^2 \nabla^2 G(r_1, \sigma^2) \quad (30) \end{aligned}$$

where $s = 1 + \alpha$. The last equality follows from the diffusion equation $\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G$. This allows the convolution (28) to be written as

$$F_1 * G(r, \sigma^2) \approx F_2 * G(r_1, \sigma^2) + \alpha\sigma^2 F_2 * \nabla^2 G(r_1, \sigma^2) \quad (31)$$

The above equation is linear in s . To find s , three filtering operations need to be performed: two gaussian filtering operations and one laplacian operation. Note that the above equation expresses the well-known result that a laplacian can be approximated by a difference of gaussians.

4.2.1 Issues of Scale

Information in an image is scale dependent. There may be information present at several different scales or at only one of them. A method which does not take this into account is not likely to be robust. Thus it is desirable to solve the above equation at several different scales (σ_i). Let a set of σ_i be chosen. For each such σ_i an equation of the form (31) may be written giving the following system of equations

$$\begin{aligned} F_1 * G(r, \sigma_0^2) &\approx F_2 * G(r_1, \sigma_0^2) + \alpha\sigma_0^2 F_2 * \nabla^2 G(r_1, \sigma_0^2) \\ F_1 * G(r, \sigma_1^2) &\approx F_2 * G(r_1, \sigma_1^2) + \alpha\sigma_1^2 F_2 * \nabla^2 G(r_1, \sigma_1^2) \\ &\dots\dots\dots \\ F_1 * G(r, \sigma_i^2) &\approx F_2 * G(r_1, \sigma_i^2) + \alpha\sigma_i^2 F_2 * \nabla^2 G(r_1, \sigma_i^2) \\ &\dots\dots\dots \\ F_1 * G(r, \sigma_l^2) &\approx F_2 * G(r_1, \sigma_l^2) + \alpha\sigma_l^2 F_2 * \nabla^2 G(r_1, \sigma_l^2) \end{aligned} \quad (32)$$

This is an overconstrained set of equations in the unknown α . The redundancy offered by the overconstrained problem also makes it more robust with respect to noise.

The particular choice of the σ_i is to some extent arbitrary although some general criteria may be specified. Too small a σ_i will make the system sensitive to noise while localisation requires that σ_i not be too large. The actual values are not very crucial. In practice, a set of eight different σ_i were chosen. They were all spaced apart by half an octave (a factor of 1.4). The filter width = $8\sigma_i$ (since the filters need to range from $-4\sigma_i$ to $4\sigma_i$). The widths actually chosen were (3,5,7,10,14,20,28,40) (see also [Jones and Malik, 1992a]).

The above system of equations was cast into the following linear least squares problem

$$\Sigma_i \|F_1 + G(r, \sigma_i^2) - F_2 + G(r_1, \sigma_i^2) + \alpha \sigma_i^2 F_2 + \nabla^2 G(r_1, \sigma_i^2)\|^2 \quad (33)$$

and was solved using Singular Value Decomposition (SVD). It was found that the lowest filters (widths = 3, 5, 7) were noisy and hence they were disregarded (one reason may be that the laplacian is noisy when the filter size is small). The scale was recovered fairly accurately using the other widths (see the experimental section for details). This set of filter widths worked better than another one where 8 filters were used with their σ_i spaced apart by a factor of 1.2; presumably because with a larger variation in scale, there is more information available at multiple scales.

4.2.2 Choosing a Different Operating Point

For large s (say $s \geq 1.3$) the recovered scale sometimes tends to be poor. This is because the Taylor series approximation is good only for a small change in σ . The problem arises because in (31) the right-hand side is expanded around the same σ_i as on the left-hand side. A better approximation is obtained by expanding σ^* as close to the correct scale as possible. An example should clarify this point. Assume that the left-hand side uses $\sigma = \sigma_0$ and that the scale change s is 1.3, then it is better to expand the right-hand side around $\sigma_1 = 1.4\sigma_0$ (i.e. the half-octave step closest to the actual scale) rather than expanding at σ_0 . In this case, (31) may therefore be modified to

$$F_1 + G(r, \sigma_i^2) \approx F_2 + G(r_1, \sigma_{i+1}^2) + \alpha' \sigma_{i+1}^2 F_2 + \nabla^2 G(r_1, \sigma_{i+1}^2) \quad (34)$$

where $s = 1.4(1 + \alpha')$. Since filtering by a set of σ is already being performed for the overconstrained system no additional filtering operations are required. Again, an overconstrained system may be implemented easily. For each value of σ_i on the left-hand side of (34), expand around σ_{i+1} on the right-hand side.

A similar scheme may be implemented if the scale $s \leq 0.8$ by expanding around a σ_i which is smaller by a factor of 1.4.

A priori the σ_i around which the expansion should be done is not known since the value of the scale is not available. The solution is to expand around all three of them (i.e. σ_i , $1.4\sigma_i$ and $\sigma_i/1.4$) and then pick the correct answer to be the one which gives σ^* close to the operating point. Again an example will clarify this point. Assume that the correct scale is again 1.3 and that the three different operating points return the following values of s (1.25, 1.32, 1.18). Consistency decides the correct answer here. 1.18 is inconsistent with expanding around $\sigma/1.4$ and can be rejected. The other answers are both between 1 and 1.4 and closer to 1.4. Therefore, the appropriate operating point to pick is $1.4\sigma_i = \sigma_{i+1}$. Experimentally, this method seems to

work well. An alternative is to compare the residual error after SVD minimisation; this does not seem to work as well, partly because the different errors are not really comparable—they have different numbers of equations. Another technique that has been tried is to make all the equations into a single overconstrained system and solve it using SVD—based on the answer obtained, some of the equations may be dropped and the system resolved.

In principle the same technique can be used to expand around nonnearest neighbor operating points σ_j , $|j - i| > 1$, if the scale gets very large (or small). The range of scales to be expected depends on the application. For structure from motion, a scale change of more than 1.4 almost never happens in practice. In finding shape from texture, in principle any scale change can occur. If the surface is smooth, it is expected that there is likely to be a neighbouring texture patch whose scale change is less than 2.5. In this case one should also expand around 2.0σ and 0.5σ in addition to σ , 1.4σ and $1/1.4\sigma$. Very high scale changes are probably difficult to measure in any case because of the extreme foreshortening that this implies. We reemphasise that, apart from such inherent limitations, our approach can in principle handle large magnitude affine transforms with little approximation, whereas previous methods were limited to small transforms.

4.2.3 Ensuring that F_i is always even

The method does not work if the output of the gaussian convolution is zero (or close to zero in practice). This can happen either if the signal is weak or if the signal shows odd symmetry along any dimension.

The 1-D case was dealt with in section 4.1. Here it is shown how a function in 2-D may always be converted into an even function. One cannot consider each dimension separately for this would destroy the rotational symmetry of the gaussian. Instead, the function is decomposed into parts which are radially even $E_r F_i$ and radially odd $O_r F_i$ where

$$E_r F_i = F_i(r) + F_i(-r) \quad (35)$$

$$O_r F_i = F_i(r) - F_i(-r) \quad (36)$$

The magnitude of both functions ($|E_r F_i|$ or $|O_r F_i|$) is then taken. The resulting functions are both even and the gaussian convolution is nonzero for both. The SVD is performed on each set of these functions separately and the one with the lower error is then used (this ensures that if either the even or odd components is really small, it is ignored).

4.2.4 Finding Image Translation

Before scale can be recovered, the two patches must be aligned by finding the image translation. These can be found using traditional optical flow or displacement schemes [Anandan, 1989]. Alternatively, the residual

Table 1:

Actual	No Noise				Gaussian Noise $\sigma = 10$			Uniform Noise $(-10,10)$		
	scale s	$\frac{z_2}{z_1} * 100$	$\frac{z_2}{z_1 - 1} * 100$		scale s	$\frac{z_2}{z_1} * 100$	$\frac{z_2}{z_1 - 1} * 100$	scale s	$\frac{z_2}{z_1} * 100$	$\frac{z_2}{z_1 - 1} * 100$
1.05	1.050	0	0		1.040	1.0	20	1.040	1.0	20
1.10	1.101	0.09	0.01		1.082	1.6	18	1.098	0.2	2.0
1.15	1.158	0.7	5.3		1.152	0.1	1.3	1.148	0.1	1.3
1.20	1.213	1.1	6.5		1.198	0.2	1.0	1.192	0.7	4.0
1.40	1.408	0.6	2.0		1.415	1.1	3.8	1.427	1.9	6.8
1.60	1.639	2.4	6.5		1.630	1.9	5.0	1.591	0.6	1.5
1.80	1.855	3.1	6.9		1.838	2.1	4.8	1.816	0.9	2.0

of the SVD error can be used to localise the image translation to ± 0.5 pixels. This is done in the following manner. The first image patch is filtered with the set of gaussians. The second patch is filtered with gaussians at every pixel in a small window centered at the first patch's location and the SVD computed. That pixel for which the SVD residual is minimised is declared to be the correct image translation. Experimentally, this method was found to work satisfactorily. Note that in general no additional filtering operations are required since the filtering operations are done at every point in the image anyway.

4.3 Experimental Results

Experiments were carried out both on synthetic images as well as a pair of real images. The first synthetic image (Figure 1) shows a cosine wave generated by the equation $F_1(x, y) = 127 \cos(\omega_y y + \phi_y)$. The cosine was picked for the following interesting properties. It can be made even or odd at any point depending on the value of ϕ_y . Further, there is no information along the y direction (the so-called aperture problem). In spite of that the scale can be recovered.

For the first experiment, ϕ_y was chosen to be zero, so that the function was even. $\omega_y = 0.2$ was chosen. A second cosine function was generated using the following function $F_2(x, y) = 127 \cos(\omega_y x + \phi_y)$ (Figure 2). F_2 is rotated 90° with respect to F_1 and also scaled by the factor s . For various values of s , the scale was recovered using the seroth moment. The results are tabulated in Table 1. The experiment was repeated with noise added. First, uniform noise ranging from -10 to 10 was added to F_2 . Second, gaussian noise with a standard deviation of 10 was added to F_2 . These results are also tabulated in Table 1. Two operating points were used: σ and 1.4σ . The appropriate operating point was picked as discussed in the text.

Table 1 is to be read as follows. The first column in Table 1 is the actual scale while column 2 shows the recovered scale in the noise-free case. Two different percentage errors are tabulated and they arise from the following considerations. Assume that an object is at a depth of z_0 and after a translation T_z in the z direction,

its new depth is $z_1 = z_0 + T_z$. Then the percentage error in finding the quantity z_1/z_0 is given by $\frac{z_2}{z_1} * 100$ and this is tabulated in column 3 for the noise-free case. On the other hand, the percentage error in finding the quantity T_z/z_0 is given by $\frac{z_2}{z_1 - 1} * 100$ and this is tabulated for the noise-free case in column 4. Which of these quantities is more important? Since the depth z_0 is a priori unknown, the quantity of relevance at least in the motion case is T_z/z_0 and the corresponding percentage error is more significant. Similar values are tabulated when gaussian noise (columns 5, 6 and 7) and uniform noise (columns 8, 9, and 10) are added.

The results show that even with noise depth reconstruction effectively has an accuracy on the order of several percent. The results are excellent in the noise-free case. The percentage errors in column 3 are all less than about 3% while even in column 4 the percentage errors do not exceed 7%. Note that the method recovers scale accurately in spite of the large rotation.

With noise added, the results are as good except for the lowest scales (1.05 and 1.10, corresponding to the largest depths). These results for the lower scales might be improved by using operating points separated by ratios smaller than 1.4.

The experiment was repeated using $\phi_y = \pi/2$ for both images. The method failed because the function now becomes an odd function at the origin and thus the result of gaussian filtering is zero. However, if the function is transformed into an even function using the methods discussed in the text, the seroth moment can once again be applied and the results are similar.

The experiment was repeated with random dot images. A random dot image of size 64 by 64 was generated (Figure 3). The image was then affine transformed and smoothed using a cubic interpolation scheme. For various values of the scale factor s , the scale was recovered using the seroth moment method. The results are tabulated in Table 2. The highest error in column 4 (relative depth error) is less than 9% if the smallest scale (1.05) is ignored. Again the relative error in s (column 3) is much lower. The error is somewhat larger in this case because the program that affine transforms the image does interpolation which tends to destroy im-

Table 2:

Actual	Recovered Scale s	$\frac{s_2}{s_1} * 100$	$\frac{s_2}{s_1 - 1} * 100$
1.05	1.059	0.9	18
1.10	1.104	0.4	4.0
1.15	1.138	0.8	8.0
1.20	1.180	1.7	10
1.40	1.398	0.1	0.5
1.60	1.569	1.99	5.2
1.80	1.722	4.3	9.8

age structure. This is more serious at the lower scales.

Finally the algorithm was tested on a pair of real images from a sequence [Sawhney and Hanson, 1991]. The images were taken with a Sony ccd camera using a robot moving straight ahead. The robot moved about 1.4 ft between frames. Since the original images were taken with the intent of using a line based algorithm, most of the objects have little intensity variation in their interior. However, the posters on the back wall show some intensity variation and can therefore be used. Points 1 and 2 were picked by hand in the first image (Figure 4). The corresponding points in the second image (Figure 5) were determined using the SVD residual error. For point 1, the recovered scale was 1.07 which corresponds to a distance of $1.4/(1.07 - 1) = 20$ ft. For point 2 the recovered scale was 1.06 which corresponds to a distance of $1.4/(1.065 - 1) = 21.5$ ft. The measured distance to the back wall is 20.3 ft. The accuracy is thus within 6%.

4.4 Solving the First Moment Equation

Again for the case where the affine transform is described by $A = sR$, the first moment equation (22) may be written

$$F_1(r) * dG(r, \sigma^2)/dr = sR F_2(r_1) * dG(r_1, (s\sigma)^2)/dr_1 \quad (37)$$

The diffusion equation applied to the derivatives of the gaussian gives the following identity

$$\partial G_{r_j} / \partial \sigma = \nabla^2 G_{r_j}, \quad (38)$$

where G_{r_j} denotes the derivative of the gaussian with respect to the r_j^{th} coordinate. Using this identity, the right-hand side of (37) is expanded around σ and rewritten as

$$F_1(r) * G_{r_j}(r, \sigma^2) \approx s v R_j F_2(r_1) * [G_{r_j}(r_1, \sigma^2) + \alpha \sigma^2 \nabla^2 G_{r_j}(r_1, \sigma^2)] \quad (39)$$

where R_j is the j^{th} row of R and $s = 1 + \alpha$. Note that there are 2 such equations. This may be more conveniently written as

$$\mu_1(\sigma) = (1 + \alpha) R [\mu_2(\sigma) + \alpha \sigma^2 \xi(\sigma)], \quad (40)$$

where

$$\xi(\sigma) = F_2(r_1) * d(\nabla^2 G(r_1, \sigma^2))/dr_1 \quad (41)$$

and

$$\mu_1(\sigma) = F_1(r) * dG(r_1, \sigma^2)/dr_1. \quad (42)$$

The rotation matrix can be eliminated by taking the dot-product (i.e. the magnitude) of both sides of (40) and equating them. This gives

$$\mu_1^T \mu_1 = (1 + \alpha)^2 [\mu_2^T \mu_2 + 2\alpha \sigma^2 \mu_2^T \xi + \alpha^2 \sigma^4 \xi^T \xi]. \quad (43)$$

This is a polynomial equation in the unknown α . As before several different scales σ_i are used to give an overconstrained system. The resulting system can be solved using the Gauss-Newton technique [Gill et al., 1981]. The Gauss-Newton procedure works by linearising the system around the current estimate of the solution reducing the problem to a linear least-squares problem. Define a vector function $c(\alpha)$ where the i^{th} component is given by

$$c_i(\alpha) = \mu_{1i}^T \mu_{1i} - (1 + \alpha)^2 [\mu_{2i}^T \mu_{2i} + 2\alpha \sigma_i^2 \mu_{2i}^T \xi_i + \alpha^2 \sigma_i^4 \xi_i^T \xi_i] \quad (44)$$

Then if α_k is the current estimate of α , then $\alpha_{k+1} = \alpha_k + p_k$ where p_k is the solution of the linear least squares problem

$$\|J_k p + c_k\|_2^2, \quad (45)$$

where a quantity subscripted by k denotes that quantity evaluated at α_k and $J(\alpha)$ is the Jacobian matrix of $c(\alpha)$.

The least squares problem was solved using SVD and convergence was found to be rapid—within a couple of iterations. The method was tested on a sine-wave pattern.

In two dimensions, the rotation may be computed in the following manner. Consider (40) again. This may be rewritten as

$$\mu_1(\sigma) = Rb, \quad (46)$$

where

$$b = (1 + \alpha) [\mu_2(\sigma) + \alpha \sigma^2 \xi(\sigma)] \quad (47)$$

is a known quantity (since α is now known). Let $b = (b_1, b_2)^T$. Then (46) can be transformed into the following form

$$\mu_1(\sigma) = B\omega \quad (48)$$

where

$$B = \begin{bmatrix} b_1 & b_2 \\ b_2 & -b_1 \end{bmatrix} \quad (49)$$

$\omega = (\cos \theta, \sin \theta)$ and θ is the rotation angle. Using the identities $\cos \theta = \sqrt{1 - \sin^2 \theta}$, and $B^{-1} = -B/(\det B)$, (48) can be transformed into the following pair of equations each linear in the unknown $\sin \theta$.

$$\sqrt{1 - [(b_1 \epsilon_1 + b_2 \epsilon_2)/(\det B)]^2} = \sin \theta \quad (50)$$

$$(-b_2 \epsilon_1 + b_1 \epsilon_2)(\det B) = \sin \theta \quad (51)$$

where $\mu_1 = (\epsilon_1, \epsilon_2)^T$. Such pairs of equations can be written for every σ_i and the resulting linear system of overconstrained equations can be solved using SVD for the rotation angle θ .

5 Future Extensions

Future work includes the solution for the case of the general affine transform as well as the use of the second moment equation. Other possibilities include the automation of the process over the entire image and the detection of occlusions. Finally, the use of the affine transform to find surface orientation from both texture and motion cues will be explored.

Acknowledgements We wish to thank Al Hanson for his useful comments on early drafts of this paper. The first author also wishes to thank Harpreet Sawhney for many fruitful discussions and for his constant encouragement.

References

- [Adiv, 1985] G. Adiv. Determining 3D motion and structure from optical flows generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):384-401, 1985.
- [Anandan, 1989] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283-310, 1989.
- [Bergen et al., 1992] J.R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proc. 2nd European Conference on Computer Vision*, pages 237-252, 1992.
- [Brown and Shyvaster, 1990] L. Brown and H. Shyvaster. Surface orientation from projective foreshortening of isotropic texture autocorrelation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):584-588, 1990.
- [Campani and Verri, 1992] M. Campani and A. Verri. Motion analysis from optical flow. *Computer Vision Graphics and Image Processing: Image Understanding*, 56(12):90-107, 1992.
- [Cipolla and Blake, 1992] R. Cipolla and A. Blake. Surface orientation and time to contact from image divergence and deformation. In *Proc. 2nd European Conference on Computer Vision*, pages 187-202, 1992.
- [Garding, 1990] J. Garding. *Shape from Surface Markings*. PhD thesis, KTH, Stockholm, 1990.
- [Gill et al., 1981] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, 1981.
- [Jones and Malik, 1992a] D. G. Jones and J. Malik. A computational framework for determining stereo correspondence from a set of linear spatial filters. In *Proc. 2nd European Conference on Computer Vision*, pages 395-410, 1992.
- [Jones and Malik, 1992b] D. G. Jones and J. Malik. Determining three-dimensional shape from orientation and spatial frequency disparities. In *Proc. 2nd European Conference on Computer Vision*, pages 661-669, 1992.
- [Kanade and Kender, 1983] T. Kanade and J. R. Kender. Mapping image properties into shape constraints: Skewed symmetry, affine-transformable patterns, and the shape-from-texture paradigm. In J. Beck et al, editor, *Human and Machine Vision*, pages 237-257. Academic Press, NY, 1983.
- [Koenderink and van Doorn, 1987] J. J. Koenderink and A. J. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55:367-375, 1987.
- [Sawhney and Hanson, 1991] H. S. Sawhney and A. R. Hanson. Identification and 3D description of 'shallow' environmental structure in a sequence of images. In *Proc. Computer Vision and Pattern Recognition Conference*, pages 179-186, 1991.
- [Super and Bovik, 1992] B. J. Super and A.C. Bovik. Solution to shape-from-texture by wavelet-based measurement of local spectral moments. In *Proc. Computer Vision and Pattern Recognition Conference*, 1992.
- [Werkhoven and Koenderink, 1990] P. Werkhoven and J. J. Koenderink. Extraction of motion parallax structure in the visual system 1. *Biological Cybernetics*, 1990.



Figure 1: Cosine Image

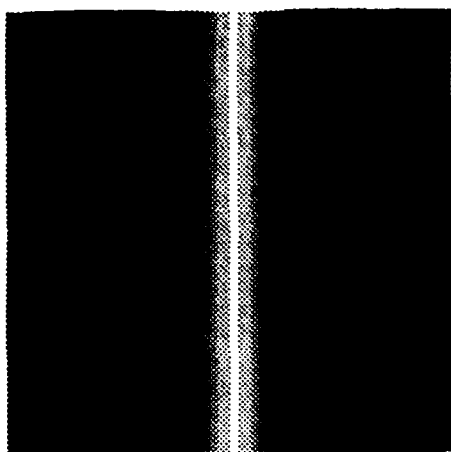


Figure 2: Scaled and Rotated Cosine Image



Figure 3: Random Dot Image

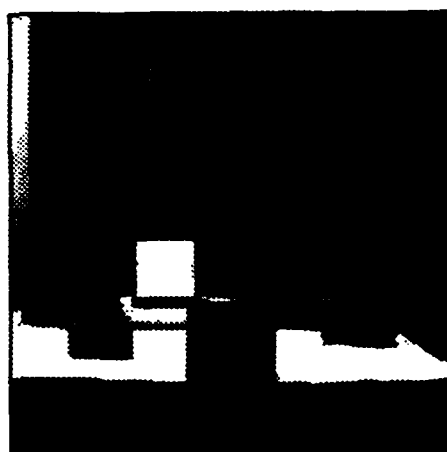


Figure 5: Real Image 2

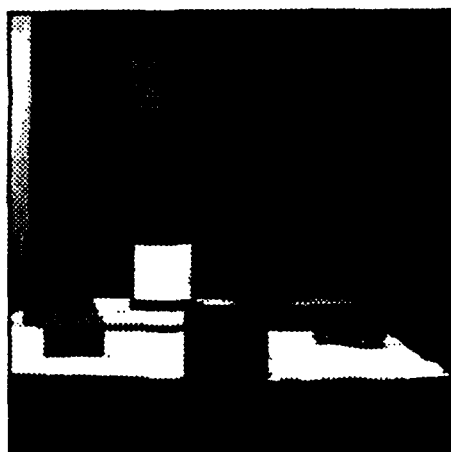


Figure 4: Real Image 1

Matching Perspective Views of Coplanar Structures using Projective Unwarping and Similarity Matching

Robert T. Collins

Department of Computer Science
University of Massachusetts
Amherst, MA. 01003
rcollins@cs.umass.edu

J. Ross Beveridge

Department of Computer Science
Colorado State University
Fort Collins, CO. 80523
ross@cs.colostate.edu

Abstract

We consider the problem of matching perspective views of coplanar structures composed of line segments. Both model-to-image and image-to-image correspondence matching are given a consistent treatment. Although these matching scenarios generally require discovery of an eight parameter projective mapping, when the horizon line of the object plane can be found in the image, using vanishing point analysis, for instance, these problems can be reduced to a simpler six parameter affine matching problem. When the intrinsic lens parameters of the camera are known, the problem further reduces to four parameter affine similarity matching.

1 Introduction

Matching is a ubiquitous problem in computer vision. Correspondence matching can be broken into two general areas: *model-to-image* matching where correspondences are determined between known 3D model features and their 2D counterparts in an image, and *image-to-image* matching where corresponding features in two images of the same scene must be identified. Fast and reliable matching techniques exist when good initial guesses of pose or camera motion are available [Beve90] or when the distance between views is small [Anan87]. What is lacking are good methods for finding matches in monocular images, formed by perspective projection, and taken from arbitrary viewpoints.

This paper examines the problem of matching coplanar structures consisting of line segments. A simple method is presented that, when applicable, allows fast and accurate matching of coplanar structures across multiple images, and of locating structures that correspond to a model consisting of significant planar patches. The main point to this paper is that the full perspective matching problem for coplanar structures

can often be reduced to a simpler four parameter affine matching problem when the horizon line of the planar structure can be determined. Given the horizon line, it is possible to transform the image to show what it would have looked like if the camera's line of sight had been perpendicular to the object plane. This process is called *rectification* in aerial photogrammetry.

2 Planar Transformations

Essentially all matching problems involve solving for both a discrete correspondence between two sets of features (model-image or image-image) and an associated transformation that maps one set of features into registration with the other. These two solutions together constitute matching: a match being a correspondence plus a transformation. For planar structures under a perspective camera model, the relevant set of transformations is the eight parameter projective transformation group [Faug88].

More restrictive transformations are worth special attention. Often these transformations are more easily computed, thus making matching easier. One such special case occurs for *frontal planes*, planar structures viewed "head-on" with the viewing direction of the camera held perpendicular to the object plane. When the intrinsic camera parameters are known, perspective mapping of a frontal plane to its appearance in the image can be described with just four affine parameters: an image rotation angle, a 2D translation vector, and an image scale [Sawh92].

2.1 Frontal Planes

Under the standard pinhole camera model, the image projection of a 3D world point (X, Y, Z) is the image point $(X/Z, Y/Z)$. In this case, the appearance of any 3D object is governed only by the relative position and orientation of the camera with respect to the object, i.e., the camera *pose*. There are 6 degrees of freedom for camera pose: three rotation angles of roll, pan and tilt, and three translations T_x , T_y and T_z . If the camera is constrained to point directly perpendicular to

*This work was funded in part by DARPA/TACOM contract DAAE07-91-C-R035 and by the RADIUS project under DARPA/Army contract TEC DACA76-92-R-0028.

an object plane, yielding a *frontal view* of the plane, its two pan and tilt angles must stay fixed. This leaves one free camera rotation about the normal of the object plane, and the three unconstrained translations, four parameters in all. The effect of camera roll on the image plane is an in-plane rotation about the origin. Translation parallel to the planar surface shows up as a 2D translation of the image. Finally, translation directly towards or away from the object plane manifests itself as a uniform change in scale of the projected image. These are precisely the effects of the four parameter affine similarity mapping. The pinhole camera projection of a frontal plane is therefore described by four affine parameters that are directly related to the physical pose of the camera with respect to the plane.

A more realistic camera model must take into account the intrinsic parameters of the camera lens. To a first approximation, lens effects are often modeled by a set of *linear* parameters including focal length, lens aspect ratio, optical center, and optical axis skew, whose combined effects can be described by a six dimensional affine mapping of the pinhole image onto the actual raster image [Horn86]. A more realistic model of the projection of a frontal plane is thus a four parameter affine mapping of the plane onto an idealized pinhole plane, followed by a six parameter affine mapping onto the actual measured image.

In summary, the perspective projection of a frontal plane is described in general by a six parameter affine transformation. When a calibrated camera is used its intrinsic lens effects are known, and can be inverted to recover the ideal pinhole projection image. After correction for lens effects, the frontal view of a plane can be described by a four parameter affine similarity mapping.

2.2 Arbitrary Orientations

For planes viewed at arbitrary orientations, the full six degrees of pose freedom may manifest themselves in the image. The two camera rotation angles, pan and tilt, not used for frontal images, are directly related to the tilt of the object plane with respect to the camera's line of sight. Under perspective projection, lines that are parallel on a tilted object plane appear to converge in the image plane, intersecting at a *vanishing point*. The locus of vanishing points of coplanar parallel lines of all orientations forms a line in the image called the *vanishing line* or *horizon line* of the plane.

For frontal planes, all parallel lines on the object plane remain parallel in the image. By convention a set of parallel lines in the image is said to intersect in a point "at infinity." When all vanishing points appear at infinity, the vanishing line passing through them is also said to be at infinity. Since a transformation is affine if and only if all parallel lines of arbitrary orientation remain parallel, it follows that the defining feature of

a frontal view of a coplanar structure is that the vanishing line of that structure appears at infinity.

Conversely, by applying a projective mapping taking the vanishing line of an image of a coplanar structure to the line at infinity, the vanishing points of all lines in the plane will now be at infinity, hence all parallel lines in the planar structure must now be parallel in the image. This implies that the new image is a frontal view of the original set of coplanar lines.

2.3 Rectification

We have seen that the vanishing line of a frontal plane appears at infinity in the image plane, and furthermore, that it is possible to recover a frontal view from the image of a tilted object plane by applying a projective transformation that maps the object's vanishing line to infinity. There is, however, a whole six-dimensional space of projective transformations that all map a given line in the image off to infinity. How to choose a "best" one is described in this section.

For a pinhole camera image, the position and orientation of the vanishing line of an object plane determines the true 3D orientation of the plane with respect to the camera's line of sight. When the equation of the vanishing line is $ax + by + c = 0$, the normal to the object plane, in camera coordinates, is

$$n = (a, b, c) / \|(a, b, c)\|. \quad (1)$$

For a frontal plane, the normal of the plane must be parallel to the Z camera axis, since the plane is perpendicular to the line of sight. If the camera could move, the image of a frontal plane could be recovered from the image of a tilted plane by merely rotating the camera to point directly towards the plane. The camera can no longer be moved physically, of course, but the image can be transformed artificially to achieve the desired 3D rotation.

Assume the unit orientation of the object plane has been determined to be n , as in equation 1, oriented into the image ($c \geq 0$). To bring this vector into coincidence with the positive Z axis requires a rotation of angle $\cos^{-1}(n \cdot (0, 0, 1))$ about the axis $n \times (0, 0, 1)$. The effects of this camera rotation on the image can be simulated by an invertible projective transformation in the image plane [Kana88]. In homogeneous coordinates,

$$k \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} E & F & a \\ F & G & b \\ -a & -b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where

$$E = \frac{a^2c + b^2}{a^2 + b^2}, F = \frac{ab(c - 1)}{a^2 + b^2}, G = \frac{a^2 + b^2c}{a^2 + b^2}.$$

The image is transformed to appear as it would have if the camera had been pointing directly towards the

object plane. The result therefore is a frontal view of the object plane, as seen by a pinhole camera, i.e. a rectified four parameter affine view.

This mapping can be used to map a vanishing line to infinity even when the intrinsic calibration parameters are not known. However, when the original image is not a pure pinhole image, the position of the vanishing line in the image can no longer be interpreted geometrically in terms of 3D plane orientation, and the resulting unwarped image will be in general some six parameter affine view of the object plane.

3 Correspondence Matching

Plane rectification forms the essence of our approach to matching perspective images of coplanar structures. The idea is to find the vanishing line of an object plane in the image by any means possible, then apply a projective transformation that maps it to the line at infinity, thereby producing an affine frontal view of the original object plane. The effect is to reduce a perspective matching problem to a simpler affine matching problem.

To search for the optimal affine map and correspondence between two sets of line segments, an efficient and robust local search algorithm is used [Beve90]. The local search matching algorithm searches the discrete space of correspondence mappings between model and image features for one that minimises a match error function. The match error depends upon the relative placement implied by the correspondence. More particularly, to compute the match error the model is placed in the scene so that the appearance of model features is most similar to the appearance of corresponding image features. The more similar the appearance the lower the match error.

To find the optimal match, probabilistic local search relies upon a combination of iterative improvement and random sampling. Iterative improvement refers to a repeated generate-and-test procedure by which the algorithm moves from an initial match to one that is locally optimal. This is done by repeatedly testing a local neighborhood of matches defined with respect to the current match. Each neighbor is a distinct correspondence mapping between model and image features. Tractable neighborhood sizes, for instance n neighbors in a space of 2^n possible matches, tend to yield tractable algorithms. However, there is an art to designing small neighborhoods that do not induce a profusion of local optima. New neighborhood definitions have been developed that are particularly well suited to shape-matching [Beve90].

Despite clever neighborhood definitions, local search can become stuck on local optima. Random sampling offers a probabilistic solution to this problem. The probability of finding the globally optimal match start-

ing from a randomly chosen initial match is analogous to the probability of getting heads when flipping an unfair coin. Even with an unfair coin it is a good bet that heads will appear at least once in a large number of throws. For instance, using a coin that only comes up heads in 1 out of 10 throws, the odds of getting heads 1 or more times in 50 throws are 99 out of 100. Similarly for local search matching, even if the probability of seeing the optimal match on a single trial is low, the probability of seeing the optimal match in a large number of trials is high.

The combination of iterative refinement and random sampling has proven to be very effective. This basic form of algorithm reliably finds excellent, and usually globally optimal, matches under difficult circumstances. The algorithm performs well even when scenes are highly cluttered and significant portions of a model instance are fragmented or missing entirely.

4 Examples

Although other methods are available (see discussion in Section 5), the results in this paper rely exclusively on vanishing point analysis for finding vanishing lines in the image. This simple approach works surprisingly well for many man-made scenes, both indoor, outdoor, and aerial. Vanishing points are found using a standard Hough transform approach [Barn83]. Each line in the image is entered into a two dimensional Hough array representing the surface of a unit hemisphere. Each image line "votes" in a great (semi)circle of accumulators, and potential vanishing points are detected as peaks in the array where several great circles intersect. For most man-made scenes, either two or three clusters will dominate the Hough array; clusters corresponding to the vanishing points of the two or three dominant line directions in the scene. Each pair of vanishing points defines a vanishing line for planes consistent with those line orientations.

At present, only a four parameter affine version of the local search matching system is implemented. We therefore needed to know the calibration parameters of the camera for each experiment. It should be stressed that only rough knowledge of the calibration parameters is generally needed to find acceptable matches. The most important parameters to determine are focal length and aspect ratio. We assumed for all our experiments that the image center was at the center of the image, and the optical X and Y axes were perpendicular (no skew) and aligned with the row and column axes of the raster image. Aspect ratio was determined from the camera manufacturer's specifications, when available, otherwise it was assumed to be one-to-one (square). The focal length for each experiment was computed as a byproduct of vanishing point analysis and *a priori* knowledge that the actual angle made by the two dominant line directions is 90 degrees

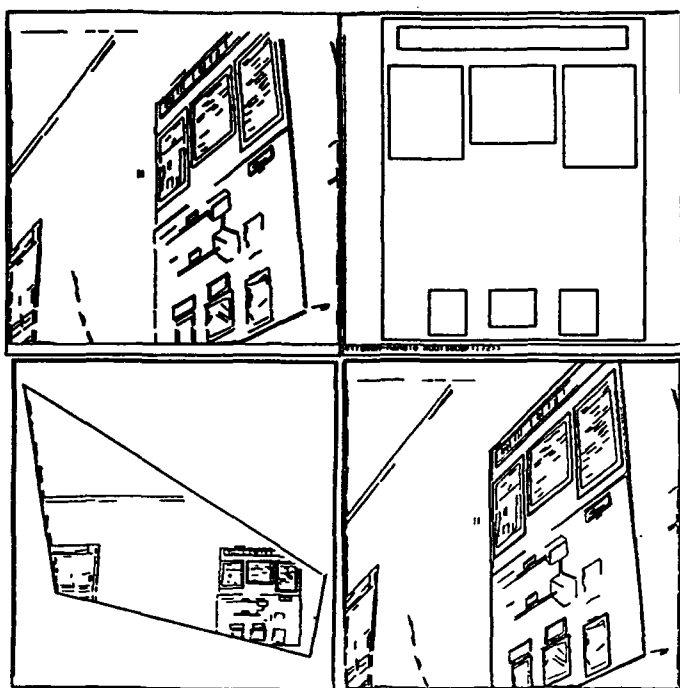


Figure 1: Model to image matching example on a poster image: a) data lines from poster image, b) poster model, c) rectified poster data lines, d) poster model registered with the image data.

[Capr90]. This focal length was chosen by finding the distance of the focal point from the image that resulted in perpendicularity of the two vectors from the (variable) focal point towards the two (fixed) vanishing points in the image.

4.1 Model to Image Matching

Figures 1a) and b) show a set of straight line segments extracted from an image of a wall poster using the Burns algorithm [Burn86], and a set of model lines to be matched to the image. The first stage in matching is to detect two clusters of lines converging to the two main vanishing points in the image, and from the resulting vanishing line rectify the image to present a frontal view of the poster (Figure 1c).

The four parameter affine match found by the local search matching algorithm yielded a set of correspondences between model lines and image lines. These correspondences were used to estimate an eight parameter planar projective transformation to bring the model lines into registration with the image data lines, using the least-squares estimation procedure of [Faug88]. Figure 1d shows the transformed model overlaid on the input image lines.

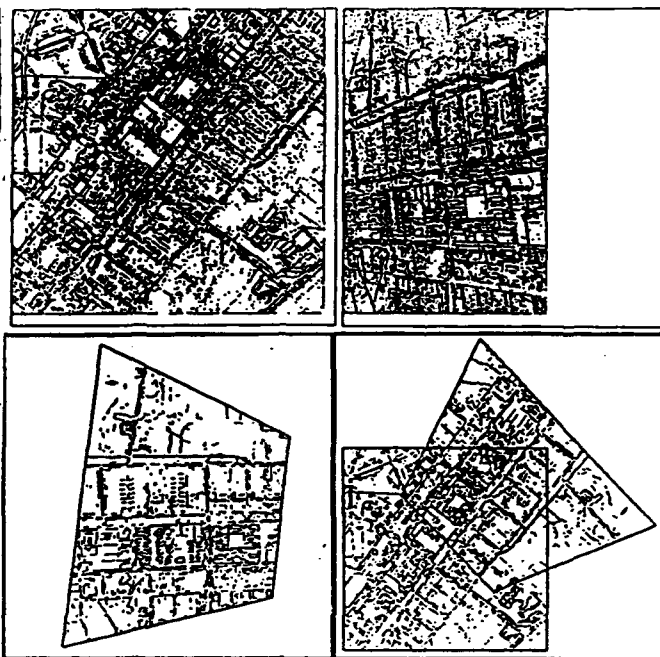


Figure 2: Image to image matching example on an aerial image: a) image lines from nadir view, b) image lines from oblique view, c) unwarped oblique view, d) registration of nadir view with unwarped oblique view.

4.2 Image to Image Matching

Because it does not rely on computing 3D object pose, the current formalism extends easily to image to image correspondence matching. In this case, both images are rectified using the techniques of the last section, and one is treated as the model while the other becomes the data to be matched. The goal is to discover a transformation that maps one set of rectified image lines into another.

When both cameras are calibrated, both images can be rectified into frontal views of the object plane. Since the mapping from one image to another can be written by inverting one transformation and composing it with the other, and since the four parameter affine group is closed under inversion and composition, the resulting image to image transformation can be described by a four parameter affine similarity map. As may be expected, when either camera is uncalibrated, the resulting transformation between unwarped views is a general six parameter affine mapping.

Figure 2 shows an example of image to image matching in the context of aerial image registration. Figures 2a) and b) show sets of extracted straight line segments from two aerial photographs. In the first image, the

camera is nearly perpendicular to the ground plane, a fact verified by vanishing point analysis, which finds two orthogonal sets of nearly parallel lines. The second image is clearly not a frontal view.¹ Figure 2c shows the image after rectification based on vanishing point analysis.

To apply the local search matching algorithm, image 1 was assumed to be the model and the unwarped lines from image 2 the data. Both line sets were filtered to only include lines greater than 100 pixels long, reducing the matching problem to 55 long lines in one image and 68 lines in the other. The best match found is displayed in Figure 2d.

5 Issues and Extensions

The domains we anticipate are scenes depicting either indoor or urban outdoor environments with much planar and parallel linear structure. Such scenes often contain lines and planes in two or three dominant directions. The approach to matching taken in this paper requires each plane to be matched separately, therefore there needs to be some way to partition lines in the image into sets belonging to planes in the world. This would be nearly impossible in monocular images, were it not for the rich structure of man-made environments, suggesting domain-specific heuristics based on corners and perpendicularity. In particular, L-junctions made of two lines from different vanishing point clusters are good candidates for coplanar corners. We are currently exploring heuristic geometric methods, as well as more formal approaches based on projective invariance, for partitioning image lines into coplanar groups.

We are also exploring other methods besides vanishing point analysis for detecting the horizon line of an object plane's image projection. Some possibilities are analysis of texture gradients [Gard91], and exploitation of the perspective properties of convex planar curves [Arns89].

When structures are present in the scene that deviate significantly from coplanarity with respect to the viewing distance, then their correspondences may not be adequately found using the above techniques. However, to the extent that some scene features are coplanar and are found, this initial set of planar correspondences provide strong constraints on the remaining features, particularly when the cameras are calibrated, in which case the relative rotation and direction of translation between the two camera positions can be computed from the planar perspective transformation [Faug88]. This reduces the problem to that of induced stereo, where point correspondences must lie along known *epipolar* lines. Even for uncalibrated

camera systems, knowledge of the perspective transformation relating the image features of a single plane in the scene constrains the positions of point features in one image to lie along epipolar lines in the other image.

In its current form, the local search affine matcher described in this paper is used for image to image feature matching simply by declaring the features in one image to be a model. This is not ideal, since the current treatment of model and image lines is not symmetric. Future work on the affine matcher may include developing a more symmetric error metric for image to image matching, and extending the range of the match transformation space to handle six parameter affine matching so that images from uncalibrated camera systems can be used.

References

- [Anan87] P. Anandan, "Measuring Visual Motion from Image Sequences," Ph.D. Thesis and COINS Tech Report 87-21, University of Massachusetts, Amherst, MA, 1987.
- [Arns89] J. Arnsperg, "Moving Towards the Horizon of a Planar Curve," *IEEE Workshop on Visual Motion*, 1989, pp. 54-59.
- [Barn83] S.T. Barnard, "Interpreting Perspective Images," *AI Journal*, Vol. 21, No. 4, November 1983, pp. 435-462.
- [Beve90] J.R. Beveridge, R. Weiss and E.M. Riseman, "Combinatorial Optimisation Applied to Variable Scale 2D Model Matching," *Proceedings IEEE International Conference on Pattern Recognition*, Atlantic City, June 1990, pp.18-23.
- [Burn86] J.B. Burns, A.R. Hanson and E.M. Riseman, "Extracting Straight Lines," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 4, July 1986, pp.425-456.
- [Capr90] B. Caprile and V. Torre, "Using Vanishing Points for Camera Calibration," *International Journal of Computer Vision*, Vol. 4, 1990, pp. 127-140.
- [Faug88] O.D. Faugeras and F. Lustman, "Motion and Structure from Motion in a Piecewise Planar Environment," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 2, 1988, pp. 485-508.
- [Gard91] J. Garding, "Shape from Surface Markings," Ph.D. dissertation, Royal Institute of Technology, S-100 44 Stockholm, Sweden, May 1991.
- [Horn86] B.K.P. Horn, *Robot Vision*, MIT Press, Cambridge, MA., 1986.
- [Kana88] K. Kanatani, "Constraints on Length and Angle," *Computer Vision, Graphics, and Image Processing*, Vol. 41, 1988, pp. 28-42.
- [Sawh92] H.S. Sawhney, Ph.D. Thesis, Computer Science Department, University of Massachusetts, Amherst, MA. 1992.

¹The term "frontal" was coined with terrestrial robotics in mind; in the aerial domain the correct term is "nadir".

Automatic Finding Of Main Roads In Aerial Images By Using Geometric - Stochastic Models and Estimation *

Meir Barzohar, David. B. Cooper

Laboratory for Engineering Man/Machine Systems

Division of Engineering, Brown University,

Providence, RI 02912

Abstract

This paper presents an *automated* approach to finding main roads in aerial images. The approach is to build geometric-probabilistic models for road image generation. Then, given an image, roads are found by map (maximum a posteriori probability) estimation. The map estimation is handled by partitioning an image into windows, realizing the estimation in each window through the use of dynamic programming, and then, starting with the windows containing high confidence estimates, using dynamic programming again to obtain global estimates of the roads present. The approach is model-based from the outset and is completely different than those appearing in the published literature. It produces two boundaries for each road, or four boundary when a mid road barrier is present.

1 Introduction

In this paper we introduce a new approach to building models for main roads in aerial images and a new computation approach to finding them. The goal is a completely automated system. The idea is that this approach could then be extended to finding other types of objects in aerial images of the ground. In recent years a number of papers have appeared in the published literature dealing with *semi-automatic* extraction of roads from aerial photos. In general a human operator gives the road starting points and the road directions at the starting points. This is a *huge* help to the road finding algorithm. This interaction has been necessary because road images can be very complicated. Examination of just the two images in this paper, Figs 6 and 8, reveals that image intensity across a road can vary noticeably. There may be a barrier running along the center of the road. Road width can vary appreciably, especially when a barrier is present. Image intensity edges along a road boundary may disappear, especially when there is a building entrance with a very small parking area alongside the road. The image intensity structure at road intersections can be very complicated. There may be cars or markings on roads, or shadows cast by buildings or trees, etc., etc.. Three major types of road finder were used: edge linkers, correlation trackers, region based followers.

Edge linkers, based on an edge operator output for magnitude and angle for each point in the image and then linking the edges according to some criteria, were used first by [6] and later by [8] and by [2].

*This work was partially supported by NSF-DARPA Grant #IRI-8905436

Correlation trackers based on the assumption that there exists a pattern or texture on the road surface was used first by [7], and later by [2] in combination with edge linkers. A region based method assuming constant intensity in the region and in the background was used first by [4] with a correlation follower.

In [1] and [3], a Bayesian approach to low-level boundary estimation and object recognition was introduced. The problem considered in this paper is the automatic extraction of main roads when road curvature, width, image intensity and edge strength can vary considerably and when a barrier along the road center may or may not be present. The approach is general, and we feel that it can be extended to handle the full range of road image variability. The approach is to build geometric-stochastic models for representing road images, and then use maximum a posteriori probability estimation for estimating the road boundaries (and other important features) in an image. The modeling approach forces the designer to model all significant phenomena, and the model is generative so that its representation power can be assessed. The map estimation provides for the most accurate road finding. Global map estimation is realized in a computationally reasonable way by using dynamic programming to search small windows to obtain initial road candidates, and then dynamic programming again with small windows in order to obtain global estimates.

2 Road Generation

2.1 Road Geometry And Internal Grey Level Model

We build a geometric-stochastic road model based on the following assumptions:

- 1) Road width variance is small and road width change is likely to be slow.
- 2) Road direction changes are likely to be slow.
- 3) Road grey level is likely to vary only slowly.
- 4) Grey level variation between road and background is likely to be large.
- 5) Roads are unlikely to be short.

A stochastic process model is built exhibiting the preceding behavior. Specifically, autoregressive processes are designed to model road center line, road width, grey level within the road, edge strength at the road boundary, and regions outside the roads and adjacent to the boundaries. We refer to these regions as background. Note that the road geometry processes are hidden, i.e., they are not observed di-

rectly in the data. For this purpose, consideration is restricted to an $N \times N$ pixel window. The stochastic processes are function of a discrete parameter i which can be thought of as time or distance, in pixels, along a horizontal or a vertical axis. As an example, consider Fig 1. The i axis here is horizontal. The road center line at i is x_i which takes values in the vertical direction. This variable is not quantized, it takes arbitrary real values. The $\{x_i\}$ process is given by (1a), where ε_{x_i} is a zero mean, white, Gaussian driving noise. This process is designed to generate a straight line if the driving noise is zero.

$$x_i = 2x_{i-1} - x_{i-2} + \varepsilon_{x_i} \quad (1)$$

$$d_i = \frac{1}{2}d_{i-1} + \frac{1}{2}d_{i-2} + \varepsilon_{d_i} \quad (2)$$

Equation (2) describes road width, where d_i is the perpendicular road width through the unquantized center point $[x_i, i]$. The perpendicular direction is measured as perpendicular to the line segment from the point $[x_{i-2}, i-2]$ to the point $[x_{i-1}, i-1]$. The stochastic processes ε_{x_i} and ε_{d_i} are independent Gaussian white noise sequences with zero means and variances σ_{x_i} and σ_{d_i} , respectively. The road obtained for the unforced solution ($\varepsilon_{x_i} = 0, \varepsilon_{d_i} = 0$) will be two parallel line as seen in Fig 1. Road boundary location are uniquely determined by the x_i and d_i . The road boundary location on the grid locations are determined as in Figure 1; where $\tilde{x}_{i,1}$ denotes the upper unquantized boundary and $\tilde{x}_{i,2}$ denotes the lower unquantized boundary.

We use a second order Markov Process to model

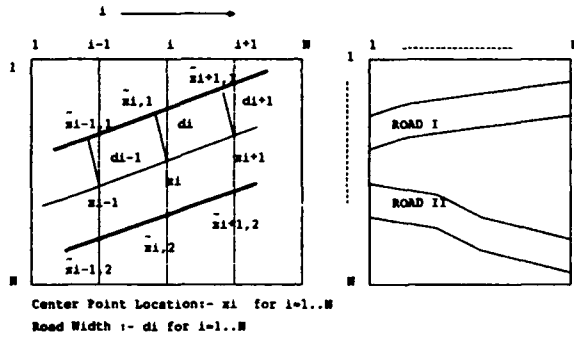


Figure 1: (A) A road in a window for unforced solution. (B) Two Roads in a Window.

the mean intensity of the image data in sequential vertical strips of the road to be consistent with feature 3 of our road model.

$$u_i = \frac{1}{2}u_{i-1} + \frac{1}{2}u_{i-2} + \varepsilon_{u_i} \quad (3)$$

The variable u_i is the mean intensity in column i of the window, and ε_{u_i} is a Gaussian white noise sequences with zero mean and variance σ_{u_i} , and is independent of the processes ε_{x_i} and ε_{d_i} . The intensity varies across the vertical direction of the road too. We model this by adding an additive white noise. Therefore, the observed picture function at the (j,i) th pixel (j th row, i th column) in the road is $y_{j,i}$, given in (4), where $n_{j,i}$ is Gaussian white noise with zero mean and variance σ .

$$y_{j,i} = u_i + n_{j,i}, \quad i = 1..N, \quad j = \tilde{x}_{i,1}.. \tilde{x}_{i,2} \quad (4)$$

Assume for now that we deal with a step edge (we also deal with the blur edge but this is more involved). The observation image intensities $y_{x_{i,2}+1,i}$ and $y_{x_{i,1}-1,i}$ immediately outside the lower and upper road boundary respectively are determined by:

$$y_{x_{i,2}+1,i} = u_i + r_2 * \theta_{i,1}, \quad i = 1..N \quad (5)$$

$$y_{x_{i,1}-1,i} = u_i + r_1 * \theta_{u,i}, \quad i = 1..N \quad (6)$$

where r_2 and r_1 are random variables, taking values ± 1 with equal probabilities, $\theta_{i,1}$ and $\theta_{u,i}$ are a white Gaussian iid sequence having some nonzero mean. The purpose of r_2 and r_1 are to model whether each of the background grey level is lighter or darker than that in the road.

Assume that the image intensities in the background regions above and below the road boundaries are modeled by different Markov processes. These are causal Gaussian AR (autoregressive) models. The lower background AR model is:

$$y_{j,i} = u_{j,i} + \sum_{k=0}^1 \sum_{l=0}^1 \beta_{kl} \times (y_{j-k,i-l} - \mu_{j-k,i-l}) + \varepsilon_{y_{j,i}} \quad (7)$$

where β_{kl} are the model parameters, $u_{j,i}$ is the mean value function, and $\varepsilon_{y_{j,i}}$ is a Gaussian white noise driving sequence with zero mean and variance $\sigma_{y_{j,i}}$. Note that this model generates image data in raster scan mode — left to right top to bottom. The conditional distribution of the process at pixel j,i given the previously generated process depends only on the three pixels immediately above and to the left of j,i . A similar model generates the background above the road, but here the process generation is left to right, bottom to top. The reason for using causal AR models is that they are computationally well suited to the dynamic programming road estimation algorithm used. With the specified road edge and background models, it is now possible to specify the probability of the background data. We are expecting to detect roads having lengths between L_{min} to L_{max} with uniform distribution. That feature is very useful in the high level processing, whereas the other features are more useful in the low level processing. In fig 2 are displayed various synthetic images which were generated by the road model described in this section.

3 Road finding as map estimation problem

Our general framework for viewing road finding is to estimate the geometry of the road by formulating the problem as map estimation. We look for that road for which $P(\text{hypothesized road model} | \text{image data})$ is a maximum. Since this posterior likelihood of a hypothesized road model given the image data can be written as

$$\frac{P(\text{hypothesized road model}, \text{image data})}{P(\text{image data})} \quad (8)$$

and the denominator is not a function of the hypothesized road model, the road model estimate can be found as that which maximizes the numerator, i.e., the joint likelihood of a hypothesized road model and the image data. The map estimation is handled by partitioning the image into windows, realizing the estimation parameters in each window

through the use of dynamic programming (details are given in [5]). Our approach also includes of design rules for starting points of a road in a window, and stopping points of a road in a window.

3.1 The complete low level processing

In the low level processing we are searching for seeds of roads. The image is divided into square windows, $N \times N$ pixels in each window, and the system searches for road candidates that fit the road model with high probability by using the dynamic programming structure. We run the road search four times, with a separate search starting at each of the four sides of the window, to pass over all the possibilities of road geometries. In Fig 3a we represent those examples. There may be more than one candidate road within a window. This is handled by searching a window for the best road, and then repeating a window search for the next best road. We do not want the second-best road to be a variation of the best road. This is handled by not permitting boundary points for the best road also to be boundary points for roads in subsequent searches. In this way, the system will find a pair of roads such as in figure 1b. The procedure is repeated until all road candidates are found. Fig 3b illustrate another example, road junction where a main road forks into two roads. In a first search starting on, the right side of the window road1 between points H, K, E, F will be found, then road2 between points L, M, F, G will be found, and road3 between points A, B, C, D will be found, by a search starting at the left side of the window. The splitting area represented by points C, D, E, F will not be found at this level. The hidden structure for every road candidate is found and observed to the high-level processing.

4 Combining road candidates

A high level processing stage is now used to extend each road candidate produced by the low level window search in order to obtain global road estimates. This is done by using shifting windows, as illustrated in Fig 4, where a new window is introduced at an end of a road candidate centered on one of the sides of the window. The best extension, of the road candidate, through the window is estimated by using the dynamic programming algorithm. This process is repeated until the stopping criterion stops the road search or until the estimated road hits the image border. In the process of extending a road through a new shifted window, the dynamic programming algorithm begins by using the last estimated state of the road and the associated road image data. Upon termination of the road search, if the length of the estimated road from the initial road candidate is greater than a threshold, the estimated road is accepted. If the length is less than the threshold, the estimated road is rejected, unless there is other supporting evidence. Supporting evidence we have used in the experiments is that if three long roads enter an intersection and the short road is adjacent to the intersection and appears to be an extension of one of them, i.e., has roughly the direction and width of one of them, then accept the short road.

5 Experimental Road Results

This section describes the results of road finding in the synthetic images (Fig2), and two different real images called, Rad1 (Fig6) and Rad2 (Fig8) that

were obtain from the Radius Program. The goal is to find the roads in the real and synthetic images using our approach. The image field is partitioned into an array of square windows (32×32). The road finder runs simultaneously within the windows to find initial road seeds in the images (section 3). It then combines all the local results to obtain the final main roads in the images by using the high-level approach. Using the low level approach *only* for the synthetic images, the results in Figure 5 were obtained. The recognized road boundary points are indicated by black dots. The recognized roads for Rad1 are indicated in (Fig7), (details are given in [5]). The recognized barriers and roads for Rad2 are indicated in (Fig9). The system starts first with recognizing the road barriers, using knowledge that the barrier intensity is lighter than the road surrounding it. The initial road barrier seeds in the image were obtain by low-level processing within the windows, and with the high-level following stage it combines the local results to obtain the final barrier (details are given in [5]). Each side of the barrier is bordered by a road; therefore, by knowing the boundaries of the barriers, the system also knows the corresponding boundaries along one side of each main road. The other boundaries along the second side of each road are estimated by using the high-level approach for road finding.

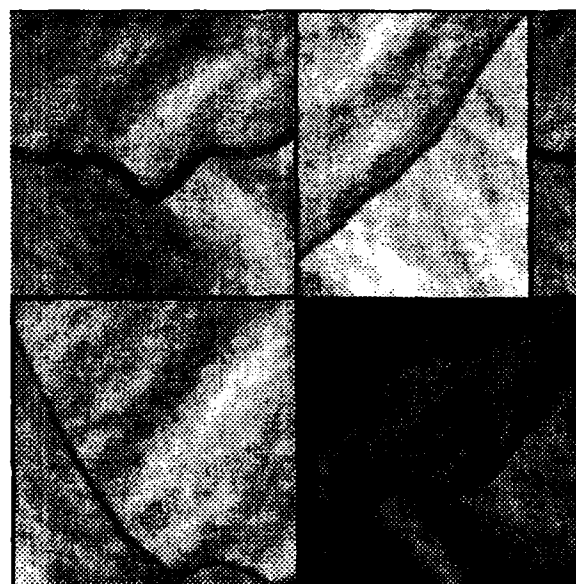


Figure 2: Various synthetic images generated by our road model.

References

- [1] D.B. Cooper. Maximum likelihood estimation of markov process blob boundaries in noisy images. *IEEE PAMI*, Oct 1979.
- [2] Jr. David M. McKeown and Jerry L. Denlinger. Cooperative methods for road tracking in aerial imagery. *CVPR*, 1988.
- [3] D.B. Cooper and F.P. Sung. Multiple-window parallel adaptive boundary finding in computer vision. *IEEE PAMI*, 1983.
- [4] Dr.-Ing. W. Kestner and Dr.-Ing. H. Kasziersak. Semiautomatic extraction of roads from aerial photographs. *Research Institute for Information Processing and Pattern Recognition*, 1978.

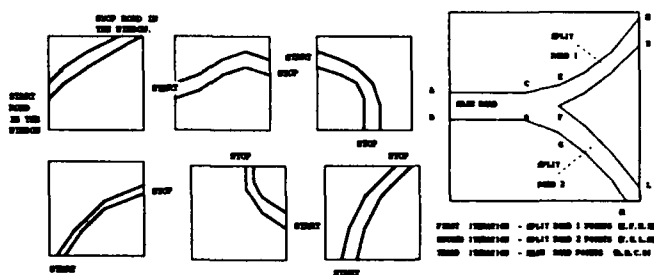


Figure 3: (a) Disparate roads with different starting and terminating side locations in a window.. (b) A Main Road split into two roads.

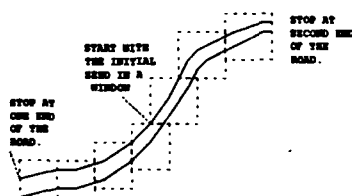


Figure 4: Following road candidate through its neighborhood shifted windows.

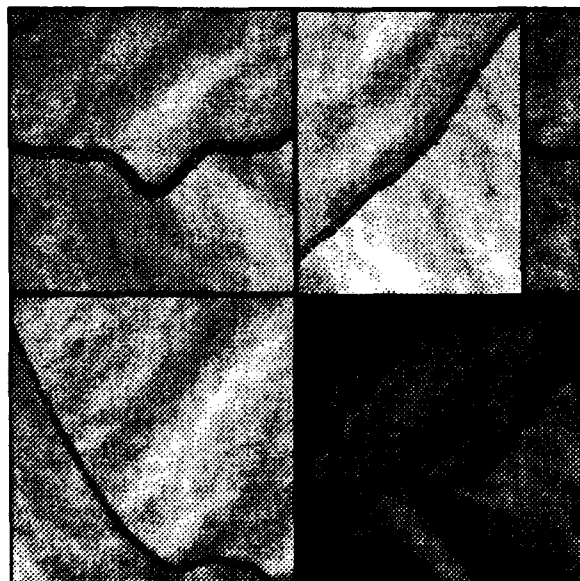


Figure 5: The recognized roads signed by black dots in Synthatic Images.

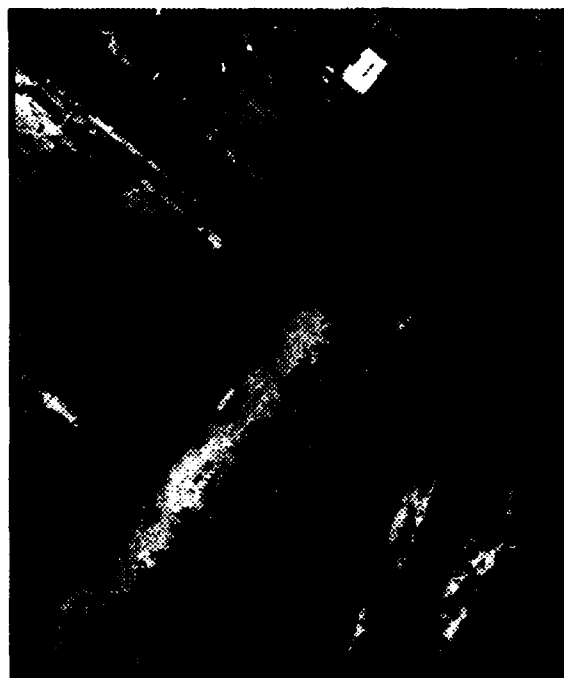


Figure 6: Real Image number 1 called "rad1" taken from the Radius program.

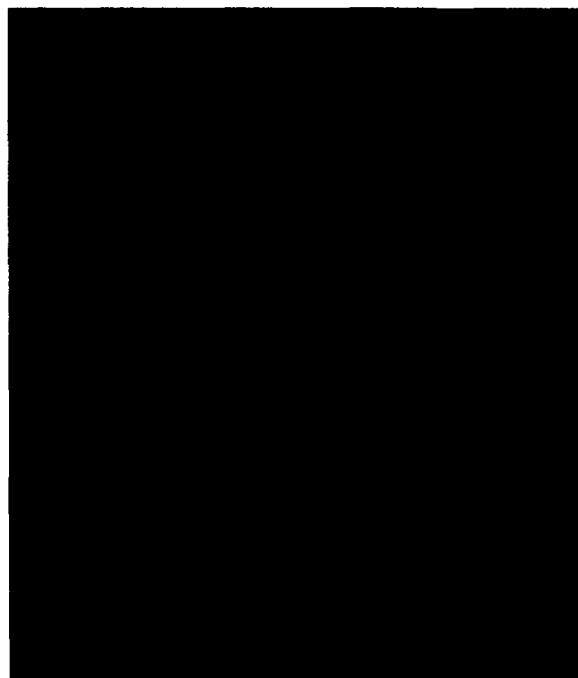


Figure 7: The recognized and estimated roads of our road finder, in image "Rad1".



Figure 8: Real Image number 2 called "Rad2" taken from the Radius program.

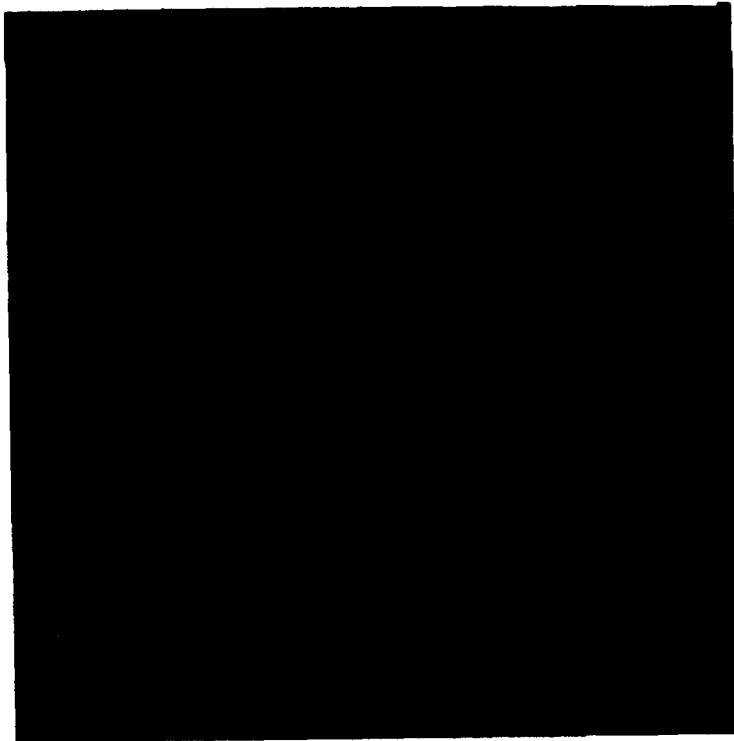


Figure 9: The recognized and estimated roads and barriers of our road finder, in image "Rad2".

- [6] M. Barzohar and D.B. Cooper. Completely Automatic Reliable Finding Of Main Roads In Aerial Images By Using Bayesian Methods. *IEEE CVPR*, July 1993.
- [6] R. Nevatia and K. R. Babu. Linear feature extraction and description. *IEEE CGIP*, 1980.
- [7] L. Quan. Road tracking and anomaly detection in aerial imagery. *Proceedings: Image Understanding Workshop*, 1978.
- [8] Z. Aviad and Jr P.D. Carnine. Road finding for road network extraction. *CMU-CS-88-157*, 1988.

Section IX

Navigation

MANIAC: A Next Generation Neurally Based Autonomous Road Follower

Todd M. Jochem

Dean A. Pomerleau

Charles E. Thorpe

The Robotics Institute, Carnegie Mellon University, Pittsburgh PA 15213

Abstract

The use of artificial neural networks in the domain of autonomous vehicle navigation has produced promising results. ALVINN [Pomerleau, 1991] has shown that a neural system can drive a vehicle reliably and safely on many different types of roads, ranging from paved paths to interstate highways. Even with these impressive results, several areas within the neural paradigm for autonomous road following still need to be addressed. These include transparent navigation between roads of different type, simultaneous use of different sensors, and generalization to road types which the neural system has never seen. The system presented here addresses these issues with a modular neural architecture which uses pretrained ALVINN networks and a connectionist superstructure to robustly drive on many different types of roads.

1. Introduction

ALVINN (Autonomous Land Vehicle In A Neural Network) [Pomerleau, 1992] has shown that neural techniques hold much promise for the field of autonomous road following. Using simple color image preprocessing to create a grayscale input image and a 3 layer neural network architecture consisting of 960 input units, 4 hidden units, and 50 output units, ALVINN can quickly learn, using back-propagation, the correct mapping from input image to output steering direction. See Figure 1. This steering direction can then be used to control our testbed vehicles, the Navlab 1 [Thorpe, 1991] and a converted U.S. Army HMMWV called the Navlab 2.

ALVINN has many characteristics which make it desirable as a robust, general purpose road following system. They include:

- ALVINN learns the features that are

required for driving on the particular road type for which it is trained.

- ALVINN is computationally simple.
- ALVINN learns features that are intuitively plausible when viewed by a human.
- ALVINN has been shown to work in a variety of situations.

These features make ALVINN an excellent candidate as the building block of a neural system which can overcome some of the problems which limit its use. The major problem this research addresses is ALVINN's lack of ability to learn features which would allow the system to drive on road types other than that on which it was trained. In addition to overcoming this problem, the system must meet the current needs of the autonomous vehicle community which include:

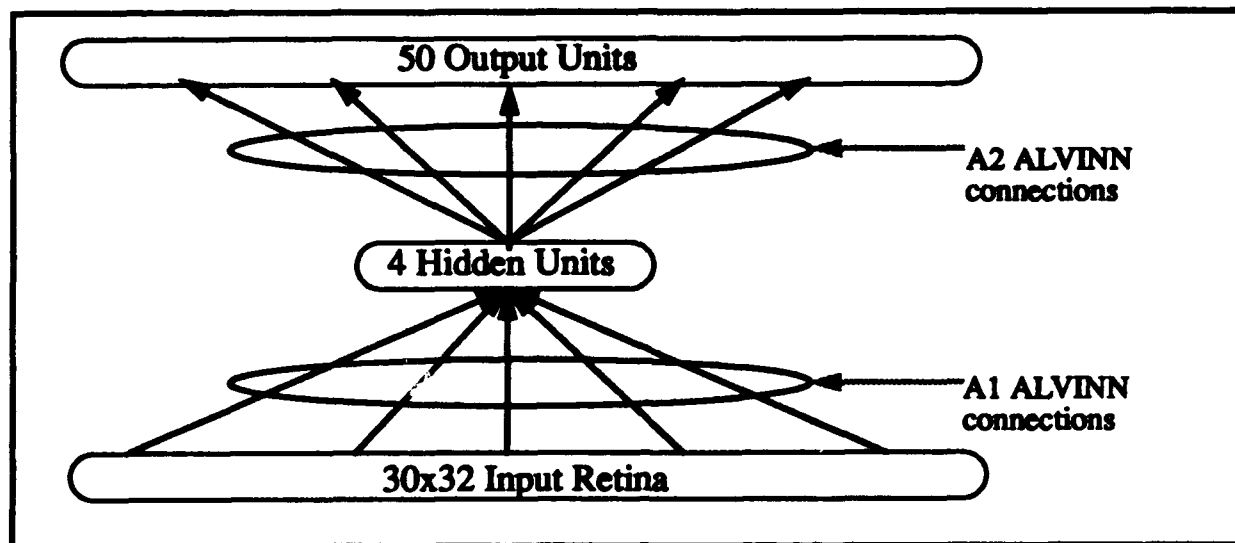


Figure 1: ALVINN network architecture.

- The ability to robustly and transparently navigate between many different road types.
- Graceful performance degradation.
- The ability to incorporate many different sensors which can lead to a much wider range of operating conditions.

From these requirements we have begun developing a modular neural system, called MANIAC for Multiple ALVINN Networks In Autonomous Control. MANIAC is composed of several ALVINN networks, each trained for a single road type that is expected to be encountered during driving. See Figure 1. This system will allow for transparent navigation between roads of different types by using these pretrained ALVINN networks along with a connectionist integrating superstructure. Our hope is that the superstructure will learn to *combine* data from each of the ALVINN networks and not simply *select* the best one. Additionally, this system may be able to achieve better performance than a single ALVINN network because of the extra data available from the different ALVINN networks.

2. The MANIAC System

The MANIAC system consists of multiple ALVINN networks, each of which has been pretrained for a particular road type. They

serve as road feature detectors. Output from each of the ALVINN networks is combined into one vector which is placed on the input units of the MANIAC network. The output from the ALVINN networks can be taken from either their output or hidden units. We have found that using activation levels from hidden units provides better generalization results and have conducted all of our experiments with this connectivity. The MANIAC system is trained off-line using the back-propagation learning algorithm [Rumelhart, 1986] on image/steering direction pairs stored from prior ALVINN training sessions.

2.1. MANIAC Network Architecture

The architecture of a MANIAC system which incorporates multiple ALVINN networks consists of a 30x32 input unit retina which is connected to two or more sets of four hidden units. (The M1 connections in Figure 2.) This hidden layer is connected to a second hidden layer by the M2 connections. The second hidden layer contains four units for every ALVINN network that the system is integrating. Finally, the second hidden layer is connected to an output layer of 50 units through the M3 connections. All units in a particular layer are fully connected to the units in the layer below it and use the hyperbolic tangent function as their activation function. Also, a bias unit with constant activation of 1.0 is connected to every hidden and output unit. The architecture of a

MANIAC system incorporating two ALVINN networks is shown in Figure 2.

The topology of the input retina and M1 connections of MANIAC system is identical to that of the A1 connection topology of an ALVINN network. See Figure 1. This allows us to incorporate an entire MANIAC system into one compact network because the A1 connection weights can be directly loaded onto the M1 connections for a particular set of first layer hidden units of the MANIAC network. Simulating the entire MANIAC system, then, does not entail data transfer from ALVINN hidden units to MANIAC input units, but only a basic forward propagation through the network.

It is the A1 connection weights of the ALVINN network that extract vital features from the input image for accurate driving. So in addition to allowing easy implementation of the MANIAC network, the network topology of the M1 connections allows us to capture important weight information in the MANIAC system that the ALVINN hidden units have learned. These features can be interpreted graphically in two dimensional views of the A1 connection weight values. Typically, a net-

work trained for one lane roads learns a matched filter that looks for the road body, while a network trained on multi-lane roads is sensitive to painted lines and shoulders.

2.2. Training the MANIAC Network

To train the MANIAC network, stored image/steering direction pairs from ALVINN training runs are collated into a large training sequence. These pairs consist of a preprocessed 30x32 image which has been shifted and rotated to create multiple views of the original image along with the appropriate steering direction as derived by monitoring the human driver during ALVINN training. See [Pomerleau, 92] for an in-depth discussion of the image preprocessing and transformation techniques. After collation, the sequence of pairs is randomly permuted so that all exemplars of a particular road type are not seen consecutively. The current size of this training sequence for a two ALVINN MANIAC network is 600. If additional ALVINN networks are used, 300 images per new ALVINN network are added to the training sequence. This sequence is stored for use in our neural network simulator.

Next, weights on each of the connections in

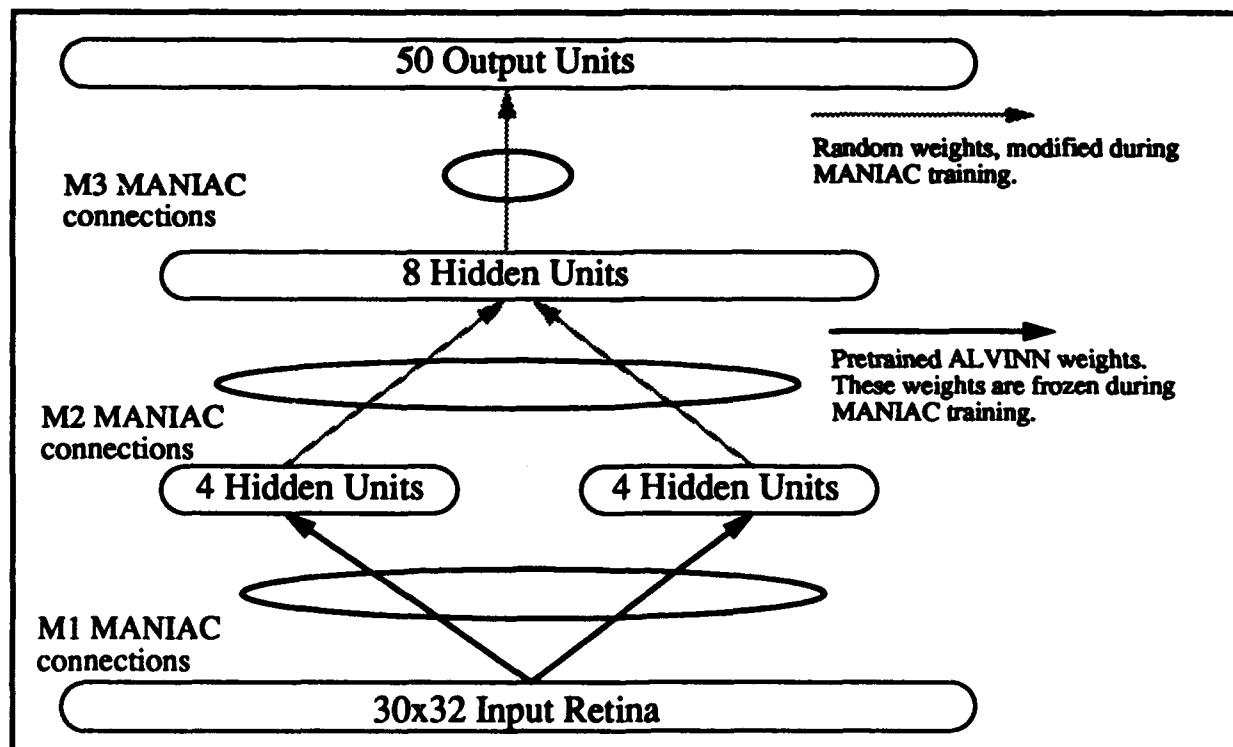


Figure 2: MANIAC network built using two ALVINN networks.

the MANIAC network must be initialized. Because the MANIAC M1 connections consist of precomputed ALVINN A1 connection weights, they must be loaded from stored weight files. After this is done, the M2 and M3 connection weights in the MANIAC network are randomized. This weight set is then ready for use as the initial starting point for learning.

To do the actual training, the network architecture along with the weight set created as discussed in the previous paragraph and the stored training sequence, are loaded into our neural network simulator. Because the MANIAC M1 connection weights are actually the pretrained ALVINN weights who serve as feature detectors, the M1 connections are frozen so that no modification during training can occur to them. See Figure 2.

Initially, training is done using small learning and momentum rates. These values are used for 10 epochs. At this point they are increased (approximately doubled) for the remainder of training. This technique seems to prevent the network from getting stuck in local minima and is an adaption of a technique used in ALVINN training.

The back-propagation learning algorithm is used to train the network. The stored images are placed on the input units of the MANIAC network while a gaussian peak of activation is centered at the correct steering direction on the 50 output units of the network. After about 60 epochs, the network has converged to an acceptable state and its weights are saved. This takes approximately 10 minutes on a Sun Sparcstation 2.

It should be noted that MANIAC uses the same output vector representation as ALVINN. This allows the output of the MANIAC network to easily be compared with that of ALVINN for quantitative study and also allows for the use of existing software in the MANIAC-vehicle interface.

2.3. Simulating the MANIAC network

Once the network has been trained, we use it in our existing neural network road following system to produce output steering directions at approximately 10 Hz.

3. Results

Empirical results of a MANIAC system composed of two ALVINN networks have been encouraging. For this system, one ALVINN network was trained to drive the vehicle on a one lane path while the other learned to drive on a two lane, lined, city street. The resultant MANIAC network was able to drive on both of these road types satisfactorily.

To determine more quantitative results, image/steering direction pairs from the same two road types as well as from a four lane, lined, city street were captured. See Figure 3. Using these stored images, ALVINN networks were trained in the lab to drive on the one lane paved path and the two lane, lined city street. Also, a MANIAC network integrating the same two ALVINN networks was trained. The results of these experiments are summarized in Table 1. In Table 1 the columns represent the average error per test image for a particular road type and the rows represent the type of network that is being used. The errors computed are of two types, SSD error and Output Peak error. SSD error is the sum of squared differences error while Output Peak error is the absolute distance between the position of the gaussian peak in the desired output activation and the peak in the actual output activation. SSD error can be thought of as a measure of the network's ability to accurately reproduce the target vector while Output Peak error is a measure of the ability of the network to produce the correct steering direction.

The initial comparison to notice in the table is that the ALVINN network trained for a particular road type always performs significantly better (> 50%) than the ALVINN network trained for the other road type when presented test images of the type of road on which it is trained. This is to be expected. Also notice that the *single* MANIAC network, which has been trained to respond properly to *both* road types, typically compares well to the correct ALVINN network (within 11% in all cases). As mentioned earlier, this amount of error is acceptable to properly drive the vehicle.

The case of the four lane road is unique in that neither of the ALVINN networks nor the

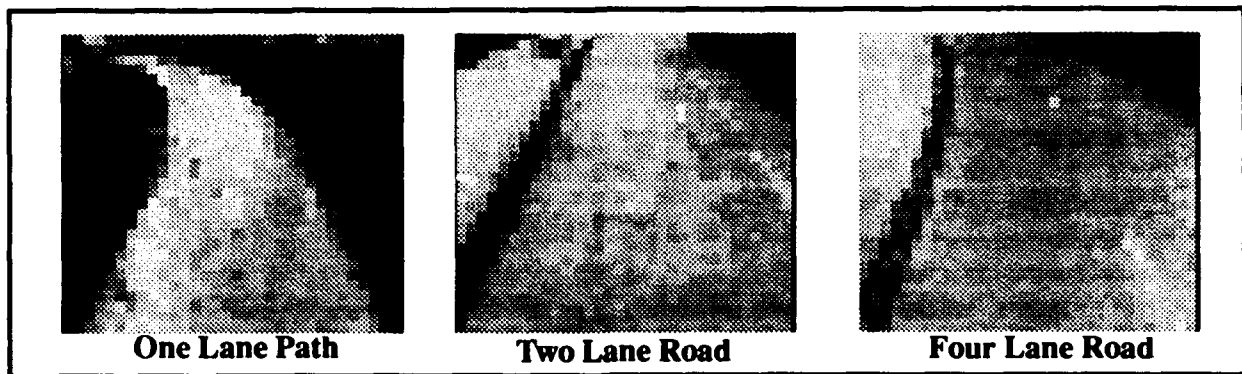


Figure 3: Typical road input images.

MANIAC network saw a road of this type. In this case, the response of the one lane path ALVINN network is nearly identical to when it was presented a two lane, lined, city street. Because this type of network typically responds to the body of the road and the fact that the two and four lane roads are both significantly wider than the one lane path, i.e. have a larger body area, this response was expected. A more interesting response is that of the two lane road ALVINN network. It seems to respond better to the four lane road images than it does to the two lane road test images. A possible explanation of why this is occurring can be seen in Figure 3. The four lane road and the two lane road look almost identical. One slight difference, though, is that the contrast of the road/offroad boundary is slightly higher in the four lane road case than it is in the two lane road case. This difference could help the network localize the road better, and because we want the vehicle to drive in the

left lane, close to the yellow line, the correct output is identical to the two lane road case.

The most interesting result, though, is that when presented with four lane road images, the MANIAC network actually performs better than either the one lane path ALVINN network or the two lane road ALVINN network. In both the prior cases, the MANIAC network performed slightly worse than the best ALVINN network for a particular road. This could imply that the MANIAC network is using information from both networks to create a reasonable steering direction at its output. This will be discussed more in the following section.

4. Discussion

A central idea that this research is trying to examine is that of improving performance and making connectionist systems more robust by using multiple networks - some of which

	One Lane Path		Two Lane Road		Four Lane Road	
	SSD	Output Peak	SSD	Output Peak	SSD	Output Peak
One Lane Path ALVINN	5.913	2.045	5.570	2.228	5.469	2.225
Two Lane Road ALVINN	11.360	3.076	3.621	1.375	1.287	0.823
MANIAC	6.263	2.167	3.907	1.532	1.243	0.774

Table 1: The average output error of ALVINN and MANIAC systems are shown for a variety of road types using two different metrics. The lower the value in the table, the better the accuracy of the network.

might be producing incorrect results. In our system the key point to notice is that although a particular ALVINN network may not be able to drive accurately in a given situation, its hidden units still detect useful features in the input image. For example, consider an ALVINN network that was trained to drive on a two lane, lined road. The features that it learns are important for accurate driving are the lines on the road and the road/non-road division. Now present this network with a paved, unlined bike path. The ALVINN network will respond in its output vector with two steering direction peaks. The reason for this is that one of the features that the network is looking for in the input image is the delineation between road and non-road. Because this occurs at two places in the image of the paved bike path, the feature detecting hidden units produce a response which indicates that the road/non-road edge is present at two locations. If these hidden unit activations were allowed to propagate to the output of the network, the characteristic two peak response would appear. Although in reality this is the incorrect response, it is a *consistent* response to this input stimulus. A similar scenario holds for other ALVINN networks given input images of road types for which they haven't been trained. Because the response of particular ALVINN network is consistent when presented with similar images, the MANIAC network can use this 'extra' data to produce a correct, perhaps better, steering direction than a single ALVINN network. It is possible that this is what is happening in the case of MANIAC driving better on the four lane road than either of the ALVINN networks.

5. Future Work

There are many directions this research can take but perhaps the most interesting is that of developing *self-training* systems. In the current implementation of the MANIAC system, ALVINN networks must be trained separately on their respective roads types and then the MANIAC system must be trained using stored exemplars from the ALVINN training runs. If a new ALVINN network is added to the system, MANIAC must be retrained. It would be

desirable to have a system that, when given initial or new ALVINN networks, created its own training exemplars and was able to automatically learn the correct MANIAC network weights. Creating training exemplars from existing network weights is essentially the network inversion problem. Techniques such as those developed by [Linden, 1989] may provide clues of how to do this one to many mapping that can create an input exemplar from an output target. It can be argued that this task is extremely difficult, even impossible, due to the high dimensionality of most networks, but perhaps it is worth taking a hard look at implementing some network inversion techniques because of the benefits that can be obtained by having self training modular neural networks.

Another area in which modular neural systems such as MANIAC may be useful is that of incorporating information from different sources. An example of this idea is to use MANIAC as a framework in which to add sensing modalities other than video. In addition to a video camera, our testbed vehicle, the Navlab 2, is equipped with an infrared camera and two laser rangefinders. If these devices can be used as input to ALVINN-like systems which produce a steering angle as output, it is reasonable to assume that a training technique similar to the one used in the current video-only MANIAC system will result in a network which will be robust in all of the component network domains. This could lead to highly robust autonomous systems which could operate in a variety of situations in which current systems fail. Driving with the same system in both daylight and at night is an example. In this scenario video images provide sufficient information to drive in the daytime but at night sensors such as infrared cameras would be necessary. The infrared cameras need not go unused in the day though, as their output would provide addition information to the modular network.

In addition to the previous areas of work, there is much to be done with developing systems which can allocate their resources and group relevant features together. It has been shown that modular neural networks can learn to allocate their resources to match a given problem,

such as locating and identifying objects in an input retina [Jacobs, 1990], while the cascade correlation algorithm provides a way to produce appropriately sized networks. [Fahlman, 1990] By using similar techniques in a MANIAC-like system, the need to pretrain ALVINN networks would be eliminated. It is not clear, though, how new information would be incorporated into this type of system once it has been trained.

6. Conclusion

This research has focused on developing a modular neural system which can transparently navigate different road types by incorporating knowledge stored in pretrained networks. Initial results from the autonomous navigation domain are promising. Although the system is simplistic, it provides a starting point from which we can explore many different areas of the connectionist paradigm such as self-training modular networks and network resource allocation. In addition to these areas, autonomous navigation tasks such as multimodal perception can be studied.

7. Acknowledgments

This research was partly sponsored by DARPA, under contracts "Perception for Outdoor Navigation" (contract number DACA76-89-C-0014, monitored by the US Army Topographic Engineering Center) and "Unmanned Ground Vehicle System" (contract number DAAE07-90-C-R059, monitored by TACOM) as well as a DARPA Research Assistantship in Parallel Processing administered by the Institute for Advanced Computer Studies, University of Maryland. Many thanks also go to the Semiautonomous Surrogate Vehicle group at Martin Marietta, Denver, where this research began.

8. References

- [Fahlman, 1990] Fahlman, S. E. and Lebiere, C., "The Cascade-Correlation Learning Architecture", Technical Report CMU-CS-90-100.
- [Jacobs, 1990] Jacobs, R. A., Jordan, M. I., and Barto, A. G., "Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks", COINS Technical Report 90-27.
- [Linden, 1989] Linden, A. and Kindermann, J., "Inversion of Multilayer Nets", in *Proceedings of the First International Joint Conference on Neural Networks*, Washington D.C.
- [Pomerleau, 1991] Pomerleau, D.A. "Efficient Training of Artificial Neural Networks for Autonomous Navigation", *Neural Computation* 3:1, Terrence Sejnowski (Ed).
- [Pomerleau, 1992] Pomerleau, D. A. *Neural Network Perception for Mobile Robot Guidance*. Ph.D. dissertation, Carnegie Mellon University, February, 1992.
- [Rumelhart, 1986] Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986) "Learning Internal Representations by Error Propagation" in Rumelhart, D.E. and McClelland, J.L. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press.
- [Thorpe, 1991] Thorpe, C., Hebert, M., Kanade, T., and Shafer, S. *Toward Autonomous Driving: The CMU Navlab*. IEEE Expert, V 6 #4, August 1991.

Perceptual Aspects of Navigation*

Herbert L. Pick, Jr., Albert Yonas, Douglas Gentile,
Patricia Melendez, Douglas Wagner, and Dominick Wegesin

Center for Research in Learning

Perception, and Cognition

University of Minnesota

Minneapolis, MN 55455

Abstract

This paper summarizes several recent results from perceptual experiments that have potential relevance for image understanding methods in vision-based navigation.

1 Introduction.

Experiments with expert map users have provided insights into strategies useful in automated systems [Heinrichs *et al.*, 1992]. More recent experiments are investigating perceptual competence in extracting navigationally salient information from realistic terrain. Results are helping us to understand the relevance of human perception to the development of machine solutions for localization problems. In addition, computational analysis of these results may help in developing better navigational assists and training procedures.

2 Perception of Distance and Slope.

When experts solve difficult localization problems, they appear to describe topographic features in terms of distinctive properties such as relative size, elevation, slope, etc. These are typically specified in bipolar qualitative terms such as large or small, narrow or wide, steep or shallow. Comparison among features is common with terms such as larger, broader, and steeper. As best as we can determine, metric descriptions in terms of units of distance or angle of slope are rarely used, despite the fact that metric information is readily available on maps with a distance scale and contour lines at standard intervals.

The psychophysical literature on perception of spatial layout suggests that at least in the case of distance estimation, judgments can be quite accurate. Generally, the results of studies of perceived distance are represented in the form of a power function relating judgments of perceived distance to actual distance: judged distance = $K * (\text{actual distance})^n$, where K is a constant which depends on the scale used and n is an exponent which defines the form of the function as decelerating, linear, or accelerating. A surprising number of studies yield

results with exponents very close to unity, indicating a linear function with a high degree of "size constancy." While many of these studies were conducted in small spaces in laboratory environments or building corridors, a number were conducted in outdoor natural environments. For example, [Da Silva, 1985] obtained functions with exponents ranging on the average from 0.87 to 0.98 depending on the particular method of estimation used.

Although the studies conducted in such outdoor environments suggest impressive precision in perception of distance, the environments used have without exception been flat homogeneous spaces such as grassy fields, athletic fields, or open water. We conducted three experiments to determine if these results generalized to situations more likely to occur in actual navigation tasks. The terrain used was part of a ski slope area and a large nature park with distances ranging from 8.5m to 357 m. Our principal conclusion is that variability was very large, with the previously accepted power function model accounting for only 40% to 60% of the variance in distance judgments in some conditions.

Scatter diagrams of subjects' judgments of distance under two of the conditions used provides some idea of such variability. Figure 1 indicates actual and estimated lateral distances between two points which lay along a single line of sight under one of the viewing conditions. Figure 2 indicates actual and estimated radial distances between two points, both at approximately the same distance from the observer.

An additional experiment was carried out in flat homogeneous terrain and yielded results similar to those reported in the literature. In addition, all four experiments replicated a result often reported in the literature, that of relative underestimation of radial distance compared to lateral distance. This is often attributed to the foreshortening of visual projections of distance along the line-of-sight.

As in the case of research on perceived distance, there is considerable laboratory research on the perception of slope. However, there is almost no research on the perception of slope in natural terrain. A common observation of the laboratory research is an overestimation of the slant of surfaces from the horizontal. Of 51 such studies overestimation was reported in 31 and underestimation in only one, with the others reporting approximately veridical judgments. In the one report of per-

*This work was supported by National Science Foundation grant IRI-9196146, with partial funding from the Defense Advanced Research Projects Agency.

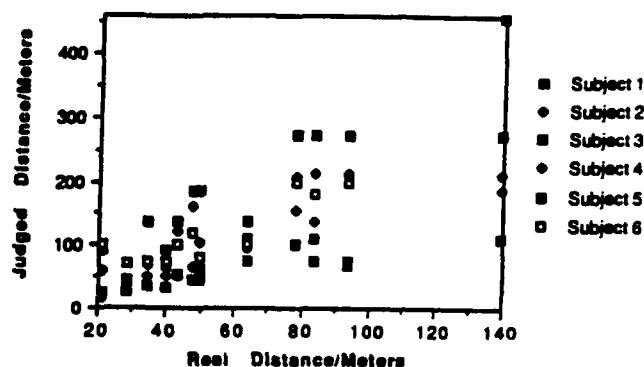


Figure 1: Estimated vs. actual lateral distances.

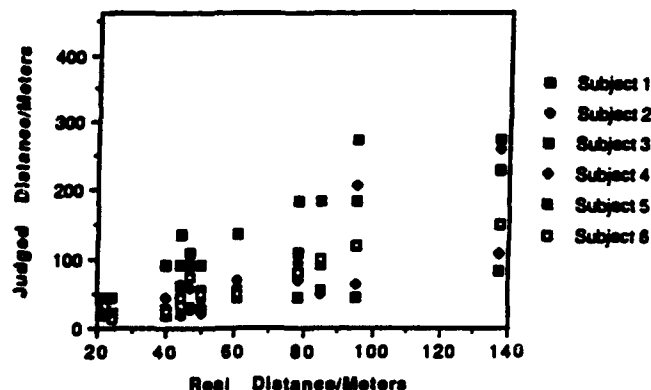


Figure 2: Estimated vs. actual radial distances.

ceived slant in natural terrain a similar observation was made [Smith and Smith, 1965]. Adults were asked to estimate the slope of a road going up a hillside. The hill, several miles away, faced the observers. All subjects grossly overestimated the slope of the hill; the measured slope was 3° but the modal estimate was 45° !

We carried out three experiments to assess the perception of slope in natural terrain. Subjects were asked to judge the average slope of the ground between two target locations. The slopes were on hillsides and level areas near a local ski area, on a river bank, and on streets in a residential environment. The particular slopes varied from horizontal to 50° . Included were hills rising in the medial plane (e.g., lateral slopes in which the slope of the hill cut across the line of sight) hills rising in the fronto-parallel plane (e.g., radial slopes which were slanting into the line-of-sight).

Results of all three experiments indicated that subjects markedly overestimated the slopes with amount of overestimation varying up to 30° . Figure 3 shows the best linear fit between actual and observed slope estimates for one experimental condition. In general the amount of overestimation was less for slopes close to horizontal. Furthermore, radial slopes tended to be overestimated to a greater degree than lateral slopes. There was also some indication that there was greater overestimation of radial slopes when they were far from the observer than when they were close to the observer. Both the greater overestimation of radial as opposed to lateral

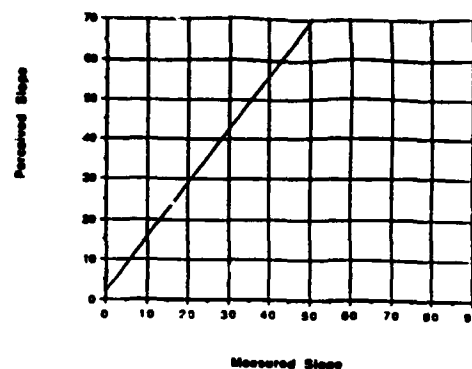


Figure 3: Actual and estimated lateral slopes.

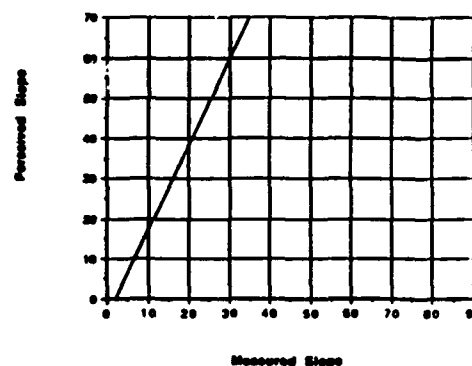


Figure 4: Actual and estimated radial slopes.

slopes and the greater overestimation of more distant radial slopes may be due to inaccuracy in radial distance judgments. If the underestimation of radial distances is greater than the underestimation of lateral distances, exactly this sort of foreshortening will occur. A similar effect may also account for the overestimation of radial slopes in laboratory studies where reduced viewing conditions are typically used.

In summary, the perception of metric characteristics of natural terrain is quite variable and subject to considerable error. Map users may know this from experience and not rely on strategies employing such information. It is not known to what degree systematic training could reduce these errors and provide map users with another useful tool.

3 Localization Based on Visual Angle.

[Levitt *et al.*, 1987] describe how the apparent order of landmarks constrains the viewpoint to a half plane defined by *landmark-pair-boundaries*. On a plane, non-linear triads of landmarks constrain the viewpoint to one of seven possible *orientation regions*. If in addition, quantitative information about the visual angles between landmarks is available, two landmarks constraint the viewpoint to a partial circle and in most cases, three landmarks uniquely determine the viewpoint. [Sutherland, 1992] extends the analysis by showing how uncertainty in viewpoint determination is related to errors in

visual angle determination and the actual positions of the landmarks used. Uncertainty in viewpoint increases with uncertainty in visual angle, though not necessarily in a linear manner. For a given accuracy in angle determination, uncertainty can vary significantly for different configurations of landmarks. For example, the uncertainty associated with three landmarks on a line and the actual viewpoint off to the side is much greater than if the center landmark were closer to the viewpoint.

The only study in the literature investigating human localization of viewpoint based on ordering of landmarks and something equivalent to landmark-pair-boundaries is that of [Peruch and Pailhous, 1986]. Subjects were given a map with 22 distinctive object locations identified. They were then given a series of trials in which they were shown a left-right ordering of a subset of the map objects, as if from an eye level view of the space of the map. The ordering of the objects was equivalent to the order that would be seen if one were at particular location on the map and the distance between the different pairs of objects corresponded to the visual angles that would be subtended by them from that location the map. The task was to find in each case this viewpoint on the map. Subjects found correct solutions in about 90% of the trials. Their behavior and comments indicated that they were using information about which landmarks were in view in the ordering and about at least large differences in visual angle subtended by pairs of object locations. The study demonstrates that, in principle, human subjects can operate on the basis of such qualitative navigation techniques. However, the large number of target locations available on any given trial makes possible localization on the basis of intersection of landmark-pair-boundaries without necessary attention to visual angle. In addition, performance on this paper-and-pencil assessment doesn't necessarily indicate what people might do in estimating outdoor locations.

An initial attempt was made to assess whether people were sensitive to the information provided by the visual angle separation of landmarks and whether their accuracy would be subject to the configuration constraints identified by [Sutherland, 1992]. Subjects were asked to locate on a map their own viewpoint, in relation to three environmental landmarks. The landmarks were buildings in a metropolitan skyline, viewed from a vantage point a mile or more across a river. The river and intervening urban clutter prevented accurate perception of even the relative distance of the individual landmarks, leaving visual angle as potentially the best information for localization. The maps were blank sheets of paper except for marks indicating the location of the three landmarks. The three buildings on the skyline were identified to a subject and then they were given a map with three marks corresponding to the buildings. The subject's task was to mark their own position on the map. In half of the configurations the landmarks were colinear. In the other half, one of the landmarks was displaced toward or away from the observer to see if this increased the precision of localization as might be implied by Sutherland's analysis. There were ten sets of the three landmarks of each type.

Results indicated, first of all, that people are rather poor at using visual angle information in this way. The average error in estimating one's position, relative to the distance to the configuration of landmarks, ranged from 30% for a configuration whose distance was on the average 1,700m to 65% for a configuration approximately 1,100m away. Thus observers were making errors of the order of 500m to 700m. For a sample of visual angles across the various configurations ranging from 14° to 70°, the visual angles associated with the estimated viewpoints differed from the actual angles in the scene by an average of almost 30%. Either the observers were not able to accurately make use of the visual angle information or they had significant difficulty in estimating the angles with precision.

Nevertheless, there was some indication that subjects could use the visual angle information. In the first place, there was a high correlation between the actual visual angles and the angles associated with the estimated viewpoints. Secondly, in spite of the large variability in performance, for eight of the ten sets of landmarks the average error was greater for the configurations with the colinear landmarks than for the corresponding sets with the medial landmark non-colinear. Performance of occasional individual subjects is very congruent with visual angle information. It is also interesting to note that in no case were landmark-pair-boundaries violated. Thus while performance is quite variable across subjects there are individual subjects who perform at a very high level and on the average subjects' performance does seem to reflect use of visual angle. Training of human attention to visual angle information for position may be feasible and is worth trying.

References

- [Da Silva, 1985] J.A. Da Silva. Scales for perceived ego-centric distance in a large open field: Comparison of three psychophysical methods. *American Journal of Psychology*, 98:119-144, 1985.
- [Heinrichs et al., 1992] M. R. Heinrichs, K. Smith, H. L. Pick, Jr., B. H. Bennett, and W. B. Thompson. Strategies for localization. In *Proc. DARPA Image Understanding Workshop*, January 1992.
- [Levitt et al., 1987] T.S. Levitt, D.T. Lawton, D.M. Chelberg, and P.C. Nelson. Qualitative navigation. In *Proc. DARPA Image Understanding Workshop*, pages 447-465, February 1987.
- [Peruch and Pailhous, 1986] P. Peruch and J. Pailhous. How do we locate ourselves on a map: A method for analyzing self location processes. *Acta Psychologica*, 61:71-88, 1986.
- [Smith and Smith, 1965] O.W. Smith and P.C. Smith. An illusion of slant in nature. *Perceptual and Motor Skills*, 20:1108, 1965.
- [Sutherland, 1992] K.T. Sutherland. Sensitivity of feature configuration in viewpoint determination. In *Proc. DARPA Image Understanding Workshop*, pages 315-319, January 1992.

Landmark Selection for Accurate Navigation*

Karen T. Sutherland
Computer Science Department
University of Minnesota
Minneapolis, MN 55455

Abstract

Robot navigation is often based on determining the relative position of visible landmarks in the environment and establishing a match between those landmarks and features on a map to determine map position. Exact measurements are seldom known, and combinations of approximate measures can lead to large errors in self-localization. The conventional approach to this problem has been to deal with the errors after they occur. We show how a careful choice of landmark configurations on which to base localization will lead to significant improvement in robot performance.

1 Introduction

A robot, using a map to navigate, is faced with a double challenge: it must be able to determine its own position on the map at any given time as well as stay close to a specified path as it moves. This is usually done by establishing a match between landmarks in the environment and map features and then using geometric relationships between the navigator and the landmarks to determine map position. Errors in position determination are a function of errors in the measurement of landmark properties and the particular method used to estimate the viewpoint. To the extent that such errors have been considered at all, it has usually been in a context of developing ways to estimate uncertainty regions associated with particular viewpoints. Our interest is in selecting landmarks in such a way that the errors which can occur are minimized, rather than having to accommodate uncertainty regions after the fact. This is particularly important when sensory processing is expensive in time or other resources, making it desirable to utilize only those landmarks which are most useful.

An unstructured outdoor environment complicates the localization process significantly. Actual distances to landmarks are often difficult or impossible to estimate, forcing solutions that depend heavily on visual bearing. Absolute bearings, registered to a map, are frequently unavailable. In this case, triangulation, which requires absolute bearings to two or more landmarks, cannot be

used. In this paper, we therefore utilize more generally applicable methods based only on relative angular measurements between landmarks.

We define the *visual angle* from a navigator to two point features as the angle formed by the rays from the navigator to each feature. A perfect estimate of visual angle between two points in three-dimensional space constrains viewpoint to a surface of revolution somewhat resembling a torus [Levitt *et al.*, 1987]. When the points can be ordered with respect to viewpoint position, the viewpoint is restricted to half the surface. It follows that, for a navigator traveling on terrain, exact knowledge of the visual angles between three points constrains viewpoint to the intersection of three surfaces and the terrain. In most cases, this intersection is a single point [Sutherland and Thompson, 1993]. For an analysis of error free localization when a two-dimensional approximation of the environment is assumed, see [Levitt *et al.*, 1987, Sugihara, 1988, Krotkov, 1989, Sutherland, 1992].

We are making the assumption in this paper that landmarks are point features and can be ordered. Mountain peaks are one example of such features, though the results hold for any point landmarks or beacons. We are also assuming that the navigator is traveling on terrain (as opposed to being in space), and that perfect measurement of visual angles to three landmarks will provide exact localization. When visual angle measure is not exact but within a given range, location is constrained to an area on the terrain. We define the *area of uncertainty* to be the area in which the navigator may self-locate for any given error range in visual angle measure. We have shown that a wise choice of landmarks will lead to a decrease in the resulting area of uncertainty and significant improvement in localization.

2 Choosing Good Landmarks

When a two-dimensional approximation of the environment is assumed, any given error bound on visual angle estimate will constrain viewpoint to a thickened ring, the thickness of the ring determined by the amount of error [Levitt *et al.*, 1987, Levitt *et al.*, 1988, Krotkov, 1989]. When three landmarks are used, any given error in estimate constrains viewpoint to the intersection of two such rings¹ [Kuipers and Levitt, 1988,

*This work was supported by National Science Foundation grant IRI-9196146, with partial funding from the Defense Advanced Research Projects Agency.

¹A third ring passing through the two landmarks lying at greatest distance from each other can be computed, but it

Sutherland, 1992]. In Figure 1a, the visual angles from the observer V to AB and BC are both 45° . The dark lines surround the area of uncertainty which represents an error bound of $\pm 13.5^\circ$ or $\pm 30\%$ in both visual angles. Although the landmarks will not always be in a straight line, the visual angles will not always be identical and the navigator will not always make the same error in estimate of each angle, the resulting area of uncertainty will always equal the intersection of these two rings.

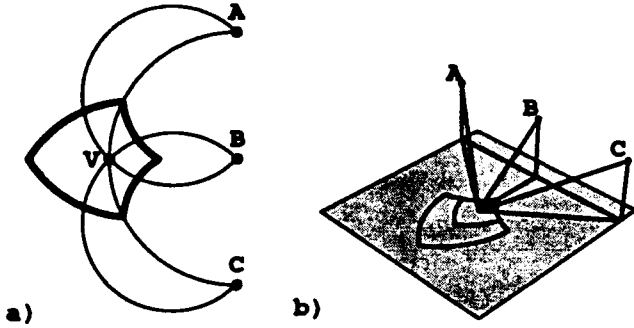


Figure 1: In a two-dimensional approximation of the environment, error in visual angle estimate to two points constrains viewpoint V to a thickened ring. When three points are used, viewpoint is constrained to (a) the intersection of two such rings. In a three-dimensional environment (b), landmark elevation affects the size of this intersection.

In a three-dimensional environment, landmarks may differ in elevation from each other as well as from the navigator. This difference will affect the size and shape of the area of uncertainty. Figure 1b shows an example of the difference that elevated landmarks can make in the area of uncertainty. The visual angles to AB and BC are both 45° . The smaller area on the plane is the area of uncertainty for planar angles of 45° and error bound of $\pm 10^\circ$ or $\pm 22\%$ if the landmark points were at the same elevation as the viewpoint. The larger area is the actual area of uncertainty for this configuration given the same error bound.

In order to make the best use of available information, the successful navigator must choose landmarks which will give the least localization error regardless of amount of error in visual angle measure. The area of uncertainty corresponding to a given visual angle and error in that visual angle varies greatly for different configurations of landmarks [Sutherland, 1992, Sutherland and Thompson, 1993]. We claim that knowledge of landmark map location and landmark viewing order, together with knowing that viewpoint is located somewhere on the map, is sufficient for choosing good configurations.

It has been shown [Levitt *et al.*, 1987, Levitt *et al.*, 1988, Kuipers and Levitt, 1988] that the lines joining pairs of landmarks divide navigation space into distinguishable areas (orientation regions). Levitt *et al.* called these lines LPB's (linear pair boundaries). When the robot passes from one orientation region to another, does not affect area size.

landmark order changes. Since the computation of the area of uncertainty is dependent on landmark order, the area will always be bounded by the orientation region formed by those three landmarks used for localization. Our algorithm begins by picking the triple of landmarks which produce the smallest orientation region on the map. (See Figure 2.)

We have previously shown [Sutherland, 1992] that the closer a configuration is to single circle (i.e., all landmarks and viewpoint on one circle), the greater the error in localization. An ongoing rule of thumb is to avoid anything near a single circle configuration. This can be done by avoiding any configuration which results in a circle which passes through all three landmarks also passing through the orientation region in which the viewpoint is required to lie. If all triples produce the same orientation region (e.g. all landmarks lie on a straight line), the most widely spaced landmarks should be chosen.

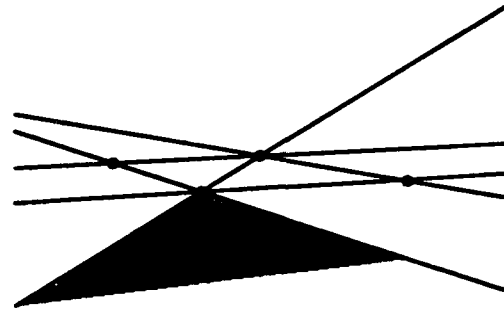


Figure 2: Lines joining the landmark points divide space into orientation regions such as the shaded area in the foreground.

Incorporating these constraints, we have developed a "goodness" function to weight configurations. It uses the locations of landmarks A, B, C and estimated viewpoint V_0 . The larger the function value, the better the configuration. Although V_0 is not necessarily the true viewpoint, our experiments have shown that this function discriminates in such a way that the best configuration to be used for localization can be determined using this estimate. Let $A = (Ax, Ay, Az)$, $B = (Bx, By, Bz)$, $C = (Cx, Cy, Cz)$, $V = (Vx, Vy, Vz)$ be the projections of the landmark points and V_0 on a horizontal plane. Let I be point of intersection of the line through V and B with the circle through A, C , and V ; L be point of intersection of the line through A and C with the line through V and B ; and $d(p, q)$ be distance between any two points p and q . (See Figure 3.)

Then:

$$G(A, B, C, V_0) = h + f$$

where

$$h = (k \left| \frac{(Az + Bz + Cz)}{3} - Vz \right| + 1)^{-1}$$

$$f = \begin{cases} \frac{2(d(V, B) - d(V, I))}{d(V, I) + \pi} * \frac{d(A, C)}{d(V, L)} & \text{if } d(V, B) \geq d(V, I) \\ \frac{d(V, I) - d(V, B)}{d(L, I)} * \frac{d(A, C)}{d(V, L)} & \text{if } d(V, L) \leq d(V, B) < d(V, I) \\ \frac{d(A, C)}{d(V, B)} & \text{if } d(V, B) < d(V, L) \end{cases}$$

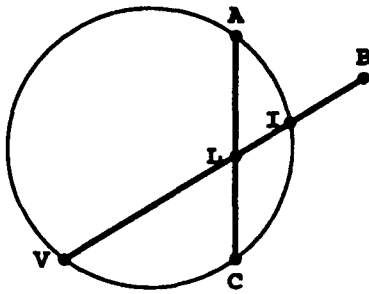


Figure 3: Simple geometric relations can be used to rank landmark configurations.

The function consists of two parts. The h function weighs the elevation of the landmarks compared to the elevation at point V_0 . It is non-negative and attains its maximum of 1 when the average elevation of the landmarks is equal to the elevation at V_0 . The constant k was set to .005 in the experiments we describe here.

The f function, also non-negative and defined piecewise, has the major effect on the goodness measure. It is based on the size of the area of uncertainty for the projected points. Note in Figure 3 that as B approaches the circle, the measure approaches zero. If B lies on line AC , the measure is the ratio $\frac{d(A,C)}{d(V,L)}$. The function increases in value as B is pulled away from the circle and estimated viewpoint. The factor of $\frac{2}{\pi}$ in the first piece of f causes this increase to occur at a rate such that when B is pulled back to the point that the area of uncertainty is the same size as for a straight line configuration, the function value is the same. The function increases in value as B moves nearer the viewpoint.

3 Experimental Results

We compared the performance of our algorithm with randomly choosing landmarks to be used for localization. All experiments were run in simulation using real topographic data. It was assumed that the navigator had a map of the area and knew map locations of points which defined both the path and the landmarks as well as the order of landmarks with respect to initial navigator location. Results for one example are shown in Figure 4 and in the sequence of frames in Figure 5. Each frame in this example represents an area approximately 18 by 12 kilometers with the lower left corner corresponding to UTM coordinates 427020E, 4497780N, southeast of Salt Lake City, UT. North is to the left of each frame, and east is toward the top. All landmarks are mountain peaks which are visible from the given path.² Identifying landmarks in large-scale space is difficult and time consuming [Thompson and Pick, 1992]. For that reason, we use a small set of landmarks. Additional landmarks can be identified if they are needed. The eight landmarks used for these trials provided 56 different combinations of ordered landmark triples.

²Landmark locations and elevations were taken from USGS 30m DEM data.

Consider two navigators moving along a path toward a goal. They have identified visible landmarks on a map and know the left to right order of those landmarks. Both begin by using their knowledge of landmark order to determine the smallest orientation region in which they are located. They use the landmarks which form that region to estimate their initial location. Those three landmarks are shown as triangles in Figure 4. The estimated location (same for both navigators) is shown by the empty square. The desired path is shown by a dotted line. The goal is marked by a star. The sequence of frames in Figure 5 show each step as the navigators progress toward the goal. A configuration of three landmarks to use for localization (triangles) is chosen. Viewpoint (empty square) is estimated and a move is made toward the next path point (line ending in solid square). The sequence on the left shows a wise choice of landmarks. Landmarks are chosen randomly in the sequence on the right.

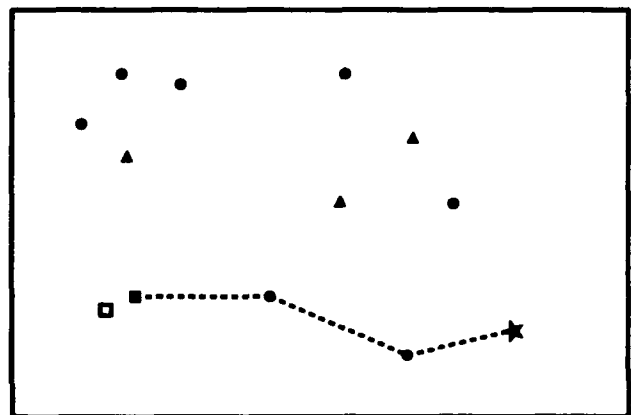


Figure 4: The eight points at the top of the figure represent the eight landmarks used for localization. Both navigators start at the solid square on the lower left. Viewpoint is estimated (empty square) using the three landmarks (triangles) which produce the smallest orientation region. Desired path is shown as a dotted line. The goal is marked by a star.

Landmarks used by the navigator on the right in the first frame are not as widely spaced as those used on the left. In addition, the center landmark lies behind (with respect to the navigator) the line joining the outer two landmarks whereas the center landmark on the left lies in front of that line. These conditions result in a larger area of uncertainty for the configuration on the right and somewhat poor localization. This error is made up for in the second frame, but a large error in estimation occurs in the last frame. The reason for this is that actual navigator location (from which the estimate was made) and the three landmarks chosen are very close to being on a single circle. The visual angles themselves in the corresponding third frames are quite similar: 28° and 45° on the left and 42° and 28° on the right.³

³Landmark elevation affects visual angle measure. That is why the sums of the angles are not equal even though the outer landmarks are the same.

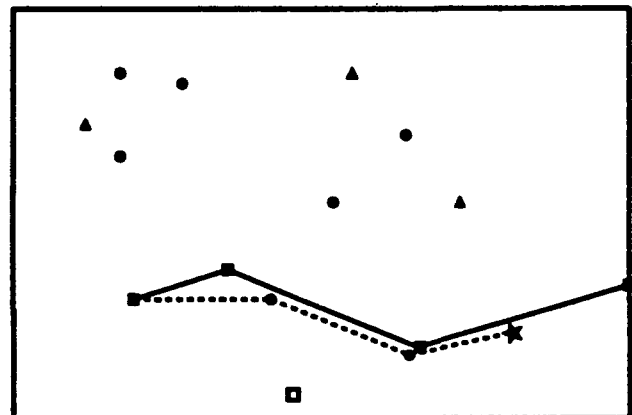
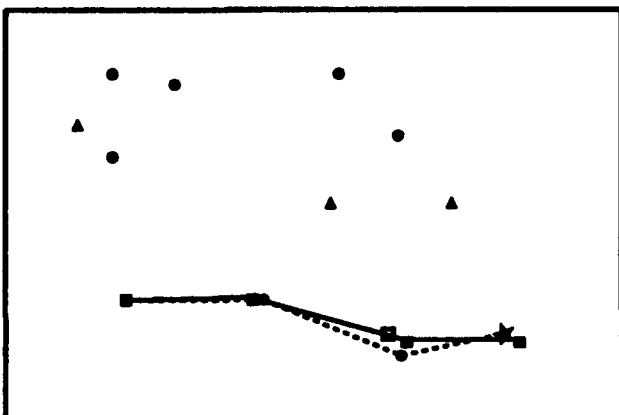
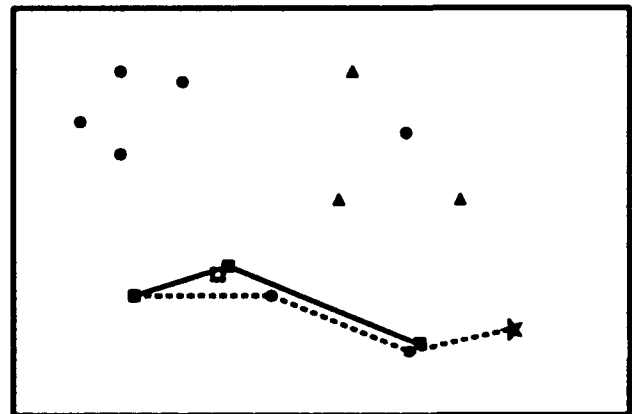
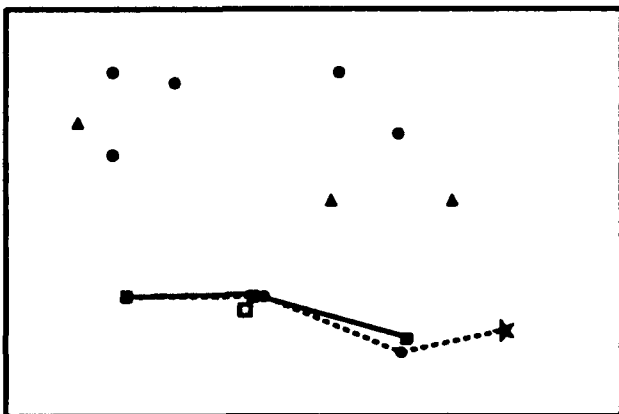
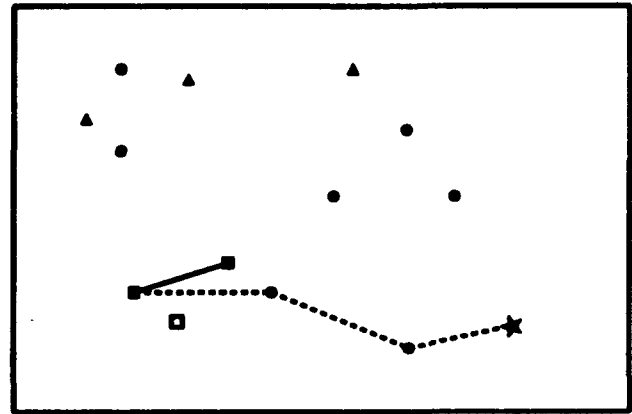
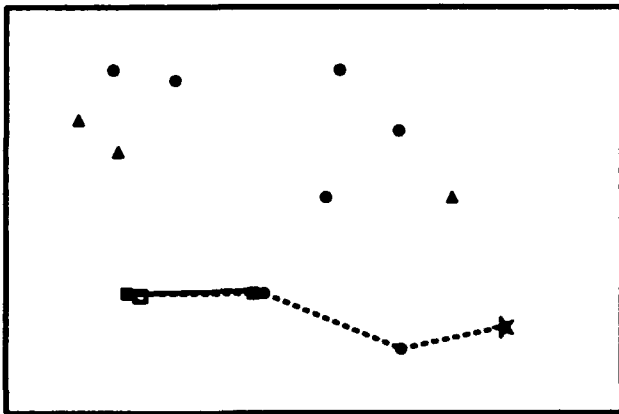


Figure 5: The sequence on the left shows the path taken by the navigator using our algorithm. The sequence on the right shows the path taken when landmarks used for localization are chosen randomly. Landmarks used for localization are shown as triangles. Desired path is a dotted line. Path taken is a solid line. Viewpoint is estimated at empty square, and navigator moves to next path point (end of solid line furthest to right).

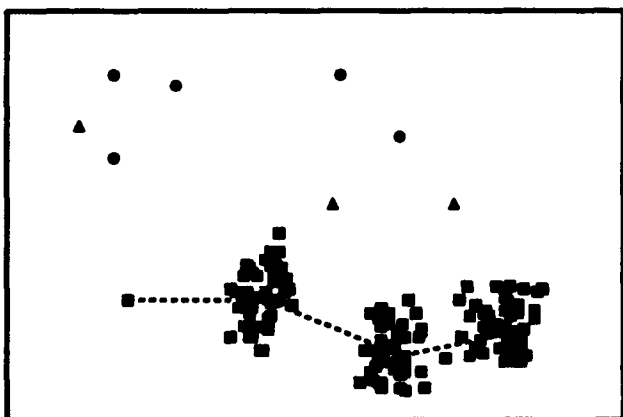
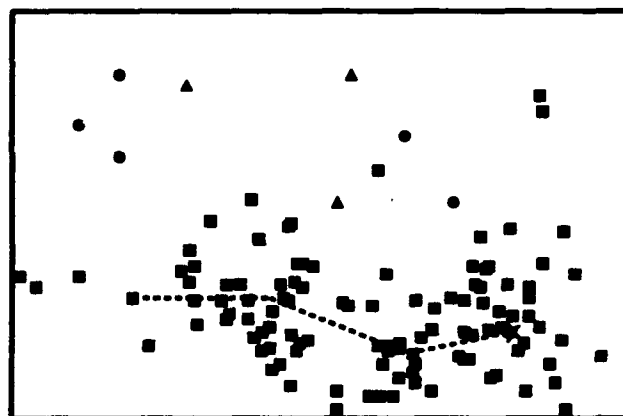
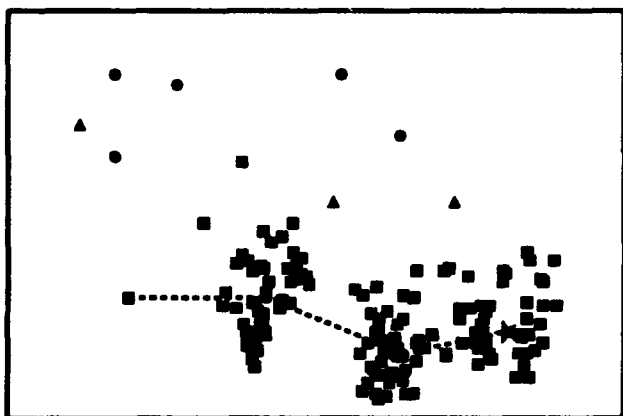
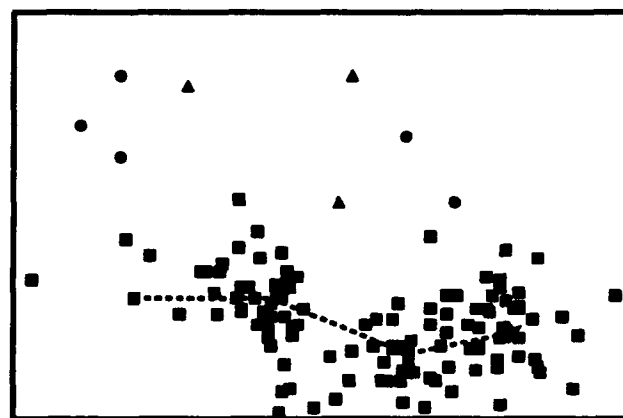
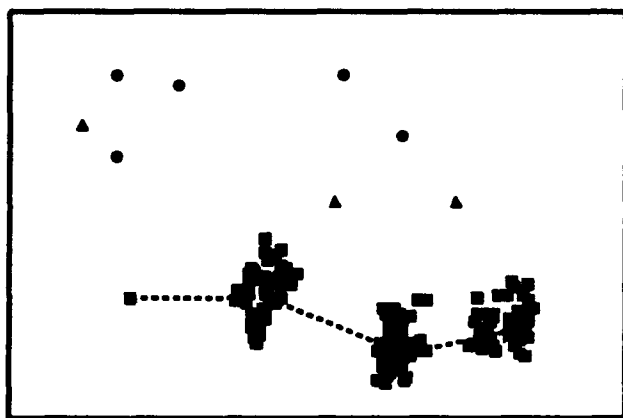


Figure 6: After fifty trials, clustering on the left shows how better localization results when landmarks are chosen wisely. Error bounds were $\pm 20\%$ in visual angle for the top pair of frames, $\pm 30\%$ in visual angle for the second pair of frames, and $\pm 20\%$ in both visual angle and direction and distance of move for the third set of frames.

Error Bounds	Wise Landmark Choice			Random Landmark Choice		
	$\pm 20\%$ Angle 0 Move	$\pm 30\%$ Angle 0 Move	$\pm 20\%$ Angle $\pm 20\%$ Move	$\pm 20\%$ Angle 0 Move	$\pm 30\%$ Angle 0 Move	$\pm 20\%$ Angle $\pm 20\%$ Move
Mean Extra Distance Traveled	344	2883	474	4273	18657	4576
Mean Distance to Path	452	513	387	1106	1227	861
Mean Distance to Goal	711	1166	769	3239	4781	3290

Table 1: Results after 100 trials. Total path length was 11352 meters. All distances have been rounded to the nearest meter.

The statistical distribution of sensor errors and whether they affect measurements in an additive, multiplicative, or non-linear manner are heavily dependent on the specific sensor technologies used. In navigation, these can range from very wide field image sensors to sensors which mechanically move to scan for landmarks. In order to illustrate our approach, we ran a simulation experiment using multiplicative error, uniformly distributed over a fixed range. Error amounts were generated using an implementation of the Wichmann-Hill algorithm [Wichmann and Hill, 1982].

The three pairs of frames in Figure 6 show navigator positions for 50 trials, assuming uniform distribution of error within $\pm 20\%$ in visual angle measure and no error in movement, error within $\pm 30\%$ in visual angle measure and no error in movement, and error within $\pm 20\%$ in both visual angle and direction and distance of move.⁴ The clustering around the path points is quite marked on the left, the result of using our algorithm to choose landmark configurations.

Table 1 gives results for all three cases after 100 trials each. Distances have been rounded to the nearest meter. "Mean Extra Distance Traveled" is the average number of meters \pm total path length that each navigator traveled. Due to the fact that paths in unstructured environments are seldom straight, total distance traveled does not necessarily reflect how well the navigator stayed on the desired path. For that reason, we also recorded distance of each path segment of the desired path to the corresponding path taken. The perpendicular distance of the midpoint of the desired path segment to the path segment taken was computed for each segment. The average of all these distances is given in the table as "Mean Distance to Path". This gives an indication of the lateral distance of each navigator to the desired path. "Mean Distance to Goal" is the average distance to the goal. The navigator which used our algorithm traveled less, remained closer to the path and ended closer to the goal than the second navigator. It is important in this type of environment that, when better localization at the goal is needed, the navigator is close enough to that goal to exploit local constraints. The navigator who chose landmarks wisely is close enough to use local constraints in all three sets of trials. It is questionable if the second navigator, averaging a minimum of two miles away from the goal, will be able to take advantage of such constraints.

⁴A point was picked from a uniform distribution within a circle of radius 20% of path segment length around the desired path point.

References

- [Krotkov, 1989] E. Krotkov. Mobile robot localization using a single image. In *Proceedings IEEE Conference on Robotics and Automation*, pages 978-983. IEEE, 1989.
- [Kuipers and Levitt, 1988] B. Kuipers and T.S. Levitt. Navigation and mapping in large-scale space. *AI Magazine*, 9(2):25-43, 1988.
- [Levitt et al., 1987] T.S. Levitt, D.T. Lawton, D.M. Chelberg, and P.C. Nelson. Qualitative navigation. In *Proc. DARPA Image Understanding Workshop*, pages 447-465, February 1987.
- [Levitt et al., 1988] T.S. Levitt, D.T. Lawton, D.M. Chelberg, K.V. Koitzsch, and J.W. Dye. Qualitative navigation II. In *Proc. DARPA Image Understanding Workshop*, pages 319-326, April 1988.
- [Sugihara, 1988] K. Sugihara. Some location problems for robot navigation using a single camera. *Computer Vision, Graphics, and Image Processing*, 42:112-129, 1988.
- [Sutherland and Thompson, 1993] K.T. Sutherland and W.B. Thompson. Inexact navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 1993.
- [Sutherland, 1992] K.T. Sutherland. Sensitivity of feature configuration in viewpoint determination. In *Proc. DARPA Image Understanding Workshop*, pages 315-319, January 1992.
- [Thompson and Pick, 1992] W.B. Thompson and H.L. Pick, Jr. Vision-based navigation. In *Proc. DARPA Image Understanding Workshop*, pages 149-152, January 1992.
- [Wichmann and Hill, 1982] B.A. Wichmann and I.D. Hill. An efficient and portable pseudo-random number generator. *Applied Statistics*, 31:188-190, 1982.

Vision-Based Localization*

William B. Thompson, Thomas C. Henderson,
Thomas L. Colvin, Lisa B. Dick, and Carolyn M. Valiquette
Department of Computer Science
University of Utah
Salt Lake City, UT 84112

Abstract

Localization based on visual landmarks requires feature extraction from views and map, matching of features between views and map, and viewpoint hypothesis generation and verification. In this paper, we describe lower-level image and map understanding procedures for extracting features and higher-level problem solving methods for establishing feature correspondences and making inferences about the viewpoint. Each of these processes, including the interaction of high-level and low-level subsystems, is demonstrated on real data.

1 Introduction.

An essential aspect of map-based navigation is the determination of an agent's current location based on sensed data from the environment. Formally, this amounts to specifying the current viewpoint in some world model coordinate system. This *localization* process has two distinct components: one involving the establishment of correspondences between aspects of the sensed data and the map or model and the other involving derivation of constraints on the viewpoint based on the correspondences that have been determined.

Correspondences can be established at the signal or feature level. Signal-level matching correlates sensed data with predictions of how the sensed data should appear. It works best when the uncertainty in the viewpoint is small and when it is relatively easy to accurately generate expected sensor data. For example, in the TERCOM and SITAN cruise missile guidance systems, a digital elevation model is matched against a downward looking, radar sensed elevation profile [Andreas *et al.*, 1978, Baird and Abramson, 1984]. Several researchers have addressed the more difficult problem of signal-based localization at or near ground level using horizontally oriented imaging systems and passive sensing. In [Ernst and Flinchbaugh, 1989], deviations between expected and observed views are determined using curve matching algorithms and a weak perspective model is used to update the position estimate. [Yacoo

and Davis, 1991] and [Talluri and Aggarwal, 1992] determine viewpoint under the assumption that viewpoint elevation is known with high precision in the reference frame of the map, a situation which dramatically reduces complexity but is unfortunately not likely to hold in practice. [Stein and Medioni, 1992] proposed an alternate method for determining viewpoint based on the observed horizon line which is similar to the characteristic view approach in object recognition.

Vision-based navigation in unstructured terrain can violate many of the assumptions used in the approaches described above. Often there is limited a priori knowledge about the viewpoint due to travel through indistinct terrain, temporary occlusion of landmark features, or errors in position updating processes. The view of the world at or near ground level is difficult to generate from map data with sufficient fidelity to allow signal-level matching. Furthermore, available digital cartographic data sets often contain inaccuracies that can cause serious problems for correlation-based analysis. For example, in one of the USGS DEMs that make up our test data, the location of the high point of a significant peak is off by over 200m. It is not surprising that most of the published work on vision-based localization from a ground-level perspective has been demonstrated only on synthetic data, where these problems do not occur.

With signal-based techniques, actual viewpoint determination is done using the same types of methods involved in photogrammetry (which solves the same problem) [Sanson, 1973, Thompson, 1958] or in alignment approaches to object recognition [Huttenlocher and Ullman, 1987, Grimson, 1990]. The principal shortcoming in both these methods is the difficulty of introducing realistic error models or effective representations of the uncertainty in viewpoint estimates.

Feature-based approaches hold the potential for avoiding many of these problems. Features are extracted independently from sensed data and maps and then matched symbolically. As a result, there is no longer a need to be able to synthesize an accurate rendition of expected sensed data. The symbolic nature of matching and viewpoint inference allows the introduction of sophisticated problem solving methods which are able to deal with issues such as ambiguity and complex error models.

In the remainder of this paper, we describe one possible approach to feature-based localization in unstruc-

*This work was supported by National Science Foundation grant IRI-9196146, with partial funding from the Defense Advanced Research Projects Agency.

tured, outdoor terrain. We outline methods for extracting terrain features from maps and image data, show how matching can be performed, and describe a collection of qualitative geometric reasoning procedures for determining viewpoint while maintaining an explicit representation of the uncertainty associated with that determination. The approach is demonstrated on a real example involving imagery obtained with a video camera and map data provided by the USGS.

2 Feature Extraction.

Three classes of entities are central to the localization process: *Terrain* is the physical layout of the land. *Maps* are geometric representations of a particular region of terrain, typically from a downward-looking perspective and possibly augmented with information about culture and/or vegetation. *Views* are visually sensed images of a particular region of terrain.

Each class of entities can be described in terms of *features*. In the case of terrain, features are commonly used geographic properties: hills, valleys, ridges, etc. These features can exist across a range of scales, specified in terms of physical extent. (We never actually deal with terrain features, only with manifestations of such features in the map and view.) In the case of maps and views, we need to distinguish between *data-level* and *terrain-level features*: Data-level features are distinctive patterns in the data (e.g., a configuration of edge fragments in a view or a locally defined topographic structure in a map). Terrain-level features are patterns of data-level features likely to correspond to some particular terrain feature. Terrain features, terrain-level map features, and terrain-level view features are distinct, even though they may have common names.

2.1 View features.

Currently, we are concentrating on those view features associated with occluding contours. Because the imagery is acquired from a horizontal perspective, these typically correspond to ridge lines. Ridge line extraction is a classical segmentation problem. The type of data we are working with, however, causes significant difficulties. Image contours corresponding to actual ridge lines should be long, connected, and relatively smooth. Except in pathological cases, they should never fold back on themselves. While this might suggest an approach which looks for "large scale" image features, things are not so simple. Contrast variations across edges that correspond to actual ridge lines can be small and of limited spatial extent. For portions of many ridge lines, contrast variations can be lacking altogether. As a result, scale-space approaches will not succeed.

Instead, we use an approach similar to [Sha'ashua and Ullman, 1988, Nevatia *et al.*, 1992]. An initial edge map is computed using a zero-crossing edge detector. Edge segments are alternately filtered to remove portions inconsistent with the geometric properties of ridge lines and augmented using properties of good continuation to account for locally indistinct ridge segments. Extraction of longer ridge lines is done using A* search [Martelli,

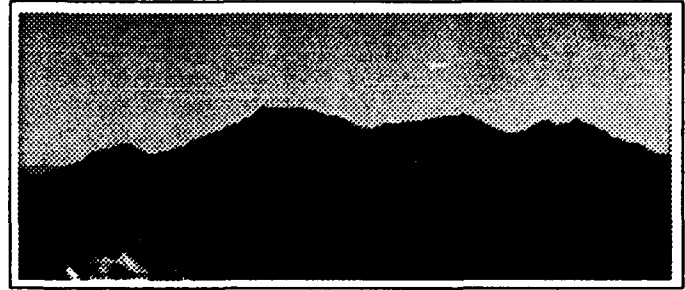


Figure 1: Original Image

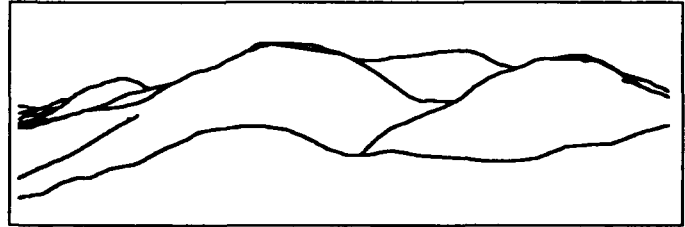


Figure 2: Actual ridge lines.

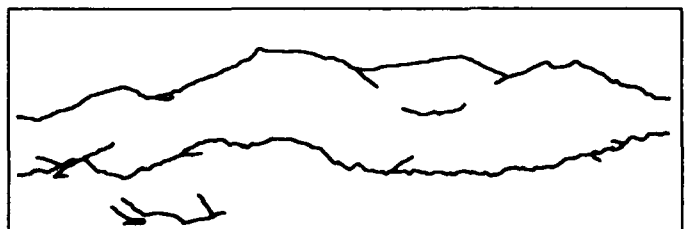
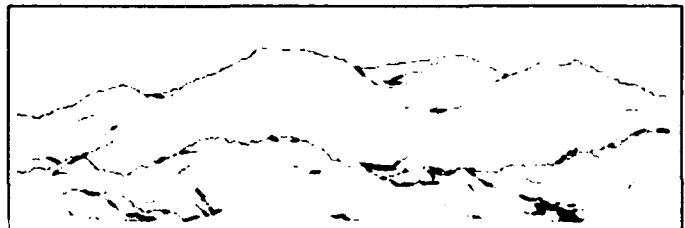
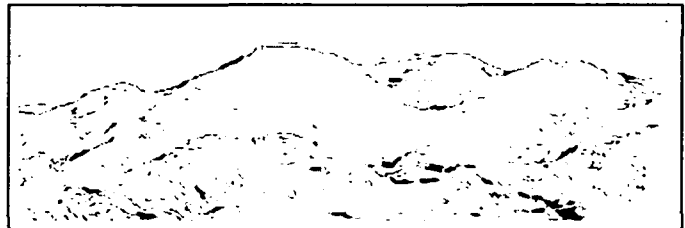
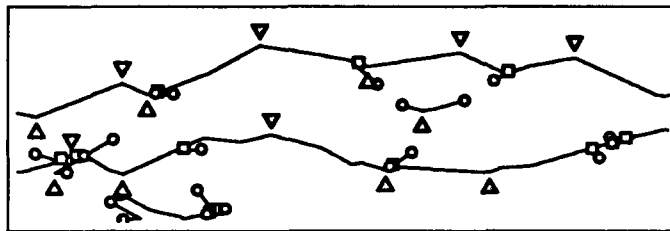


Figure 3: Edge filtering and gap filling



▽ - peaks, Δ - saddles, ◻ - junctions, ○ - endpoints

Figure 4: Extracted features

1972] which allows the specification of more global optimization criteria. This is particularly important for the horizon line, which can be difficult to find due to clouds or aerial perspective. Once these operations are completed, junctions, end points, and vertical contour extrema are located, since these often correspond to topographically relevant features.

Figure 1 shows an image of mountainous terrain. Figure 2 shows the actual ridge lines apparent from the viewpoint associated with Figure 1, as determined from DEM data. Figure 3 shows four stages in the edge filtering/gap filling process. The first frame is the output of a hysteresis thresholded zero-crossing edge detector. The next two frames show intermediate results, with filled gaps indicated by darker lines. The last frame shows the final edges. Figure 4 shows extracted line segments, peaks, saddles, T-junctions, and end points.

2.2 Map features.

The extraction of map features involves different problems than those associated with the view, but many of the processing steps are similar. (The cartographic community has done related work, but not specifically in support of localization.) Since we are operating directly on elevation data, we do not need to deal with the ambiguity associated with low-level contrast features in the view. However, we do have to find long ridge contours that may not be immediately apparent at a given scale. The analysis starts with a characterization of local surface shape of the map in terms of ridges, valleys, peaks, and saddles using the method described in [Haralick *et al.*, 1983]. Instead of resampling to produce precise ridge lines, we found it sufficient to impose thresholds when extracting ridge lines and use a thinning algorithm to extract ridge contours.

Navigationally salient ridge lines and peaks cannot be detected from an analysis of features extracted from local differential properties alone. Visually prominent ridge lines often contain broad sections where the spatial derivatives of elevation are low, resulting in a classification as flat ground and so creating breaks in the ridge lines. Local maxima in elevation may or may not correspond to visually identifiable peaks, depending on the nature of surrounding peaks and saddles. Relatively simple gap filling and filtering operations can significantly improve the utility of features extracted using local methods.

The features resulting from this process span a wide



Figure 5: Extracted peaks and ridges.



Figure 6: Extracted peaks and ridges at coarser scale.

range of spatial scales. Again, the linear nature of ridge lines limits the value of a straightforward scale-space analysis. A local analysis of ridge line junctions has proven adequate for distinguishing between dominant and subsidiary ridges. This allows the creation of a graph-like description of ridge structure, since spur ridges can in turn contain sub-spurs. Access to this hierarchy can prove significantly beneficial in feature matching. At initial stages of the matching process, only main ridges should be considered. When precise localization hypotheses are being evaluated, however, the detailed structure of the ridge line may become relevant. The hierarchical description makes it easy to avoid this level of detail unless needed.

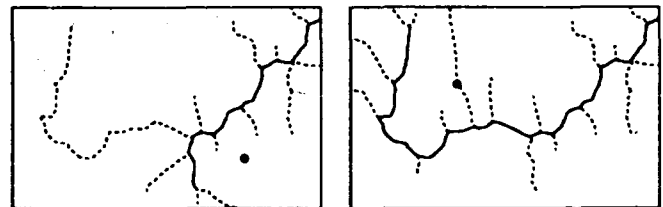


Figure 7: Alternate, viewpoint dependent ridge hierarchies.

Local analysis at times is unable to distinguish with confidence the major and minor ridges coming into a junction. Rather than being a deficiency of the approach, it may provide useful information. These situations are exactly those where there is a viewpoint dependence in the ridges which needs to be attended to. As a result, we can isolate the viewpoint dependent aspects of the representation, use with greater confidence those parts that show no obvious ambiguity, and distinguish between dominant and subsidiary features if and when hypotheses about the viewpoint are available.

Figure 5 shows significant peaks and ridges extracted at one particular spatial scale overlaid onto the corresponding contour map. Figure 6 shows peaks and ridges extracted at a coarser spatial scale. Figure 7 illustrates the hierarchical nature of extracted ridge features. "Dominant" ridge lines are often viewpoint dependent, as shown in the two parts of the figure, each with a view position indicated by a black dot.

3 Matching and Geometric Inference.

Feature-based localization involves problem solving [Heinrichs *et al.*, 1992]. The integration of symbolic problem solving with signal-level image analysis has long been a goal for many in the computer vision community. Few successful examples exist, however. In our case, we are able to effect this integration by restricting ourselves to a specific task and establishing a protocol for the interaction between high and low level analysis routines that is tailored to that task. The problem solving component of the system interacts with the feature extraction modules as if they were databases. Query and response languages were defined that make it possible to easily express relevant information about terrain features. Geometric inference is integrated in a similar manner. The result is a system in which the individual components can be constructed in a nearly independent manner, without a need to understand the details of internal representations and algorithms of other modules. Figure 8 shows the basic organization.

Overall control is determined by the high-level matching and inference system. Both top-down and bottom-up feature extraction is easily accomplished, however. For example, early in the localization process reconnaissance queries can request a general examination of map or view to determine significant features. Later, expectations can be verified in a top-down manner by generating highly constrained queries and examining whether or not any items are returned.

3.1 Matching.

One key observation arising from our study of how expert map users solve difficult localization problems is that they organize map and view features into *configurations* before attempting to match them [Pick *et al.*, in press]. Configurations are small groupings of features (typically two or three) that are close together and often satisfy particular topographic and/or geometric properties that make them distinctive. Matching configurations rather than individual features significantly reduces the combinatorics in two ways: there are fewer candidates for

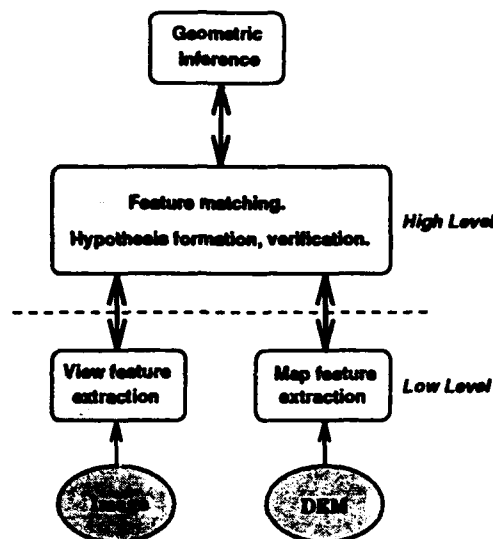


Figure 8: Interaction of high-level and low-level subsystems.

matching and each potential match involves a richer set of properties which can be evaluated for compatibility.

Complexity and possible ambiguity are further reduced by forming target configurations that are likely to be matched. Each map and view feature has a set of associated properties which constitute a geometric description of its shape and position. Particular property values such as high or sharp peaks or near level ridges are an indication that the feature is likely to be easy to find in both map and view. Specific combinations of feature properties are used to compute a set of *prominence* values, represented using a number in the range [0.0 – 1.0]. Prominence alone is a poor criterion by which to select features for forming configurations, however, since it is computed on a per-feature basis and it may turn out for a particular case that there are many features high in some particular property prominence. The *distinctiveness* of a particular prominence type is characterized by a value that is large when the population of features is such that a few have large values for the prominence in question and the rest have small values. The *saliency* of each feature property is computed by multiplying the property prominence by the property distinctiveness. Finally, the overall saliency for features is computed using a simple product rule that favors features with several highly salient prominences:

$$S_{\text{overall}}(f_i) = 1.0 - \prod_j (1.0 - S_j(f_i))$$

where $S_{\text{overall}}(f_i)$ is the overall saliency of feature (f_i) and $S_j(f_i)$ is the individual saliency of the j -th property of feature (f_i).

The formation of configurations is implemented by first sending a *reconnaissance* query to either the map or view feature extraction subsystems, requesting that individual features with high prominence be returned. These are filtered to remove all but the features with the highest overall saliency. Configurations are then formed with

a combination query to the same low-level subsystem, requesting any sets of features that contain at least one of the salient individual features and satisfy particular geometric and/or topographic properties. Any configurations that result from this process can be searched for using a similar combination query to the other low-level feature extraction subsystem.

3.2 Inference.

Localization involves the determination of viewpoint constraints based on possible correspondences between image and map features. These constraints are used in geometric reasoning operations that either hypothesize a possible viewpoint or evaluate a hypothesis by predicting additional constraints that should be satisfied. There are distinct categories of information about feature position that in turn lead to distinct constraints on viewpoint:

Absolute bearing: This is the "standard" way to solve localization problems. It requires an accurate compass registered to the map coordinate system. Determination of viewpoint is done using straightforward trigonometry.

Relative bearing: Relative bearings between three or more image features with known map positions lead to a classical "pose estimation" problem. Well established numerical techniques exist for solving such problems. [Levitt *et al.*, 1987] describe an alternate method in which only two features are considered at a time. The visual angle between the two features constrains the viewpoint to lie on a particular circle on the map. Using multiple pairs of features usually allows a unique viewpoint to be found by intersection.

Ordinal view: [Levitt *et al.*, 1987] show how ordinal position of two features (e.g., "A is left-of B") can be used to constrain the viewpoint to lie on one side of a line through the positions of A and B [Levitt *et al.*, 1987, Levitt *et al.*, 1988]. They suggest intersecting this constraint for many different pairs of features.

Exact Alignment: If two features line up along a line of sight, then the viewpoint is constrained to lie on a line connecting the two features. In almost all circumstances encountered in outdoor navigation, it is possible to determine which of the two features is more distant and as a result the viewpoint can be constrained to a half-line.

Approximate Alignment: If two features are much closer laterally (i.e., perpendicular to the line of sight) than in depth (i.e., parallel to the line of sight), then not only is the viewpoint constrained to lie on one side of a line connecting the two features, but it will be "near" this line. This constraint appears to be used with some frequency by expert map users solving real navigation problems.

Viewpoint terrain type: A locomoting agent often has more complete information about its immediate environment than it does about more distant aspects of the terrain. This information relates in a direct manner to the determination of viewpoint. (E.g., "I'm standing on a hill. Therefore, the viewpoint must be located at one of the hills found represented on the map.")

Constraints arising from the information sources listed above can be divided into two categories. All but the

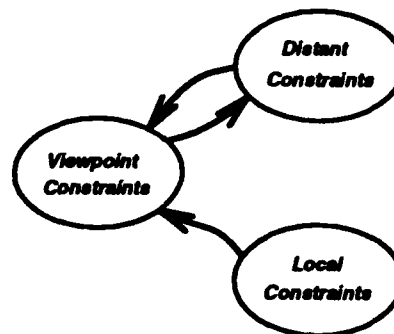


Figure 9: Reasoning about viewpoint involves interacting constraints.

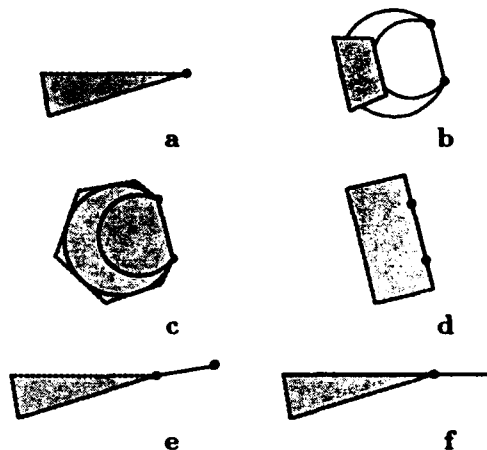


Figure 10: Distant constraints on viewpoint: a) absolute bearing, b) relative bearing if approximate depth information is available, c) relative bearing if no depth information is available, d) ordinal position, e) exact alignment, f) approximate alignment

last constitute *distant constraints*, since they are based on features distant from the viewpoint. Trigonometric relations are required to relate distant constraints to viewpoint, although qualitative as well as quantitative solutions exist. Viewpoint terrain type leads to *local constraints* which limit the viewpoint to compatible terrain features on the map. Distant and local constraints can be used in three kinds of reasoning about viewpoint (Figure 9):

Distant constraints \Rightarrow constraints on viewpoint: Map-view feature correspondences for sets of distant features can be used to determine constraints on viewpoint using geometric reasoning methods applied to any combination of the information sources described above.

Local constraints \Rightarrow constraints on viewpoint: Local constraints allow for the enumeration of possible viewpoints. Such an enumeration can be intersected with the constraint regions that usually arise from consideration of distant features.

Constraints on viewpoint \Rightarrow expectations about distant constraints: Hypotheses about viewpoint can be evaluated by examining distant features using the geometric



Figure 11: View.

reasoning methods applied to any combination of the information sources described above. Positional and/or orientational constraints on viewpoint can be exploited.

In order to implement the constraint satisfaction shown in Figure 9, geometric representations are needed for *viewpoint regions* (areas in the map corresponding to possible viewpoints), *map search regions* (map regions possibly containing terrain features visible in the view), and *view search regions* (portions of the view in which particular terrain features indicated by the map are expected to be found). A variety of representational formalisms are possible, each with advantages and disadvantages. [Sutherland and Thompson, 1993] use an analytical description of the bounding curve associated with the region in which there is any chance that the viewpoint is located. The localization example shown in section 4 uses a much simpler convex polygon representation. This provides a compact description that is efficient to manipulate. It also fits fairly well with people's intuition about the geometry involved. With a few exceptions, convex polygons have proven to be an adequate basis on which to build the geometric constraint satisfaction algorithms, though they sometimes lead to a very conservative approach such as describing the relative bearing constraint using a circle rather than a crescent (see Figure 10). Figure 10 indicates how distant constraints on viewpoint can be represented using the convex polygon approach. Similar regions have been defined for using viewpoint to develop constraints on view and map positions [Thompson, 1993].

Geometric inference is implemented using a query language similar in principle to that used to interact with the low-level feature extraction subsystems. Viewpoint regions are hypothesized or refined using a query containing a (possibly empty) current viewpoint region hypothesis, a set of corresponding map and view feature locations, and a particular inference method to use. The geometric inference subsystem applies the method to produce a viewpoint region constraint, intersects this with the initial regions supplied, and returns the result. Map search regions result from queries which specify a current viewpoint region hypothesis, a set of view feature locations, and an inference method. View search regions are obtained in an analogous manner.

4 Example.

We have demonstrated the sufficiency of the approach described above by applying it to a real example from the mountains just southeast of Salt Lake City, UT. View features were extracted from the panorama image shown in Figure 11. Note that, rather than the synthesized views commonly used in much of the reported research on outdoor localization, we are using an actual terrain image. Map features were extracted from USGS 30m

DEM data covering the equivalent of four 1:24000 7.5' quadrangles (approximately 21.4 km by 28 km), the upper half of which appears in Figures 5 and 6. The compass orientation of the view was known, but no information about viewpoint was provided other than that it was somewhere within the available map area.

The problem solving subsystem responsible for feature matching and hypothesis generation and evaluation was implemented in Lisp. The geometric inference subsystem was also implemented in Lisp and was interfaced via function calls. Both of the low-level feature extraction subsystems were implemented in C, ran on different machines than the Lisp processes, and were interfaced using simple database-like query/response techniques.

The example used four of the six high-level reasoning strategies described in [Heinrichs *et al.*, 1992]: concentrate on the view first, organize features into configurations, pursue multiple hypotheses, and evaluate hypotheses using a disconfirmation process. Two types of distant constraints on viewpoint (section 3.2) and one type of constraint for determining map search regions based on hypothesized viewpoint were used. Execution proceeded in five stages:

View reconnaissance: The view (image) was searched for significant features. The highest peak in the image stood out well above other features in overall saliency.

Form view configurations: View configurations were formed from the selected peak feature and other nearby features that were prominent. Prominence rather than saliency was used, since the configuration is more distinct than its components. Two dual-feature configurations resulted, one involving the horizontal ridge segment to the left of the peak and one involving the ridge segment to the right. In fact, the second of these was a bad choice. The ridge to the right is actually quite distant from the peak, but this was not detected by the low level image analysis routines since the corresponding T junction was not found. Simultaneous consideration of multiple hypotheses combined with the disconfirmation strategy resulted in a system tolerant of such errors.

Search for configurations in map: Configurations consisting of a high, sharp peak and a nearby horizontal ridge were searched for in the map. Three such configurations were found.

Generate initial hypotheses: Six configuration matching hypotheses were postulated (two view configurations times three map configurations). Each configuration match specified two feature matches which were used to generate a hypothesized viewpoint region using the relative bearing constraint, as shown in Figure 12.

Refine hypotheses and evaluate using disconfirmation strategy: For each hypothesis, highly salient view fea-

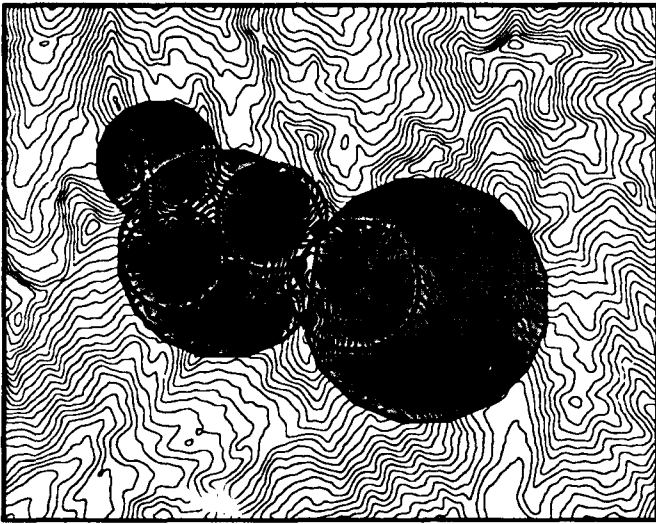


Figure 12: Viewpoint regions corresponding to the six initial hypotheses.

tures were considered in turn and searched for in the map. If exactly one map feature of the correct type was found in the expected location, a match was established and the absolute bearing constraint was used to refine the viewpoint region. If two or more map features were found, no inferences were drawn and the next view feature was processed. If no map feature was found where one was expected, the hypothesis was disconfirmed.

Figure 13 shows the refinement of the hypothesis corresponding to the actual viewpoint. Four view features were searched for in the map: the high peak mentioned previously, two other peaks towards the left of the panorama, and the long ridge line that wraps around from the right edge to the left edge of the panorama image. Three unique matches were found, involving two of the peaks and the long ridge. The remaining peak was ambiguous, with two possible peaks in the map located in positions that could plausibly correspond to the location of the view feature. On the left of the figure is shown the search for corresponding map features. The current viewpoint hypothesis is shown together with the search region predicted from the bearing to the chosen view feature. Black dots indicate map features that were found. For the first three view features considered, a unique map feature was found. On the right is shown the current viewpoint, the constraint regions associated with the map feature just found, and the intersection which becomes the refined viewpoint region. The last map search returned an ambiguous result, as can be seen by the two features present within the search region. As a result, no refinement of the viewpoint region was possible. The plot on the lower right of the figure shows the final region, with the actual viewpoint marked. The original viewpoint hypothesis had an area of approximately 1.489 km^2 . After the first absolute bearing constraint was imposed, the size of the region was down to $72,800 \text{ m}^2$. The second absolute bearing constraint left this unchanged. The final constraint reduced the area to less than $71,700 \text{ m}^2$, or about $.07 \text{ km}^2$.

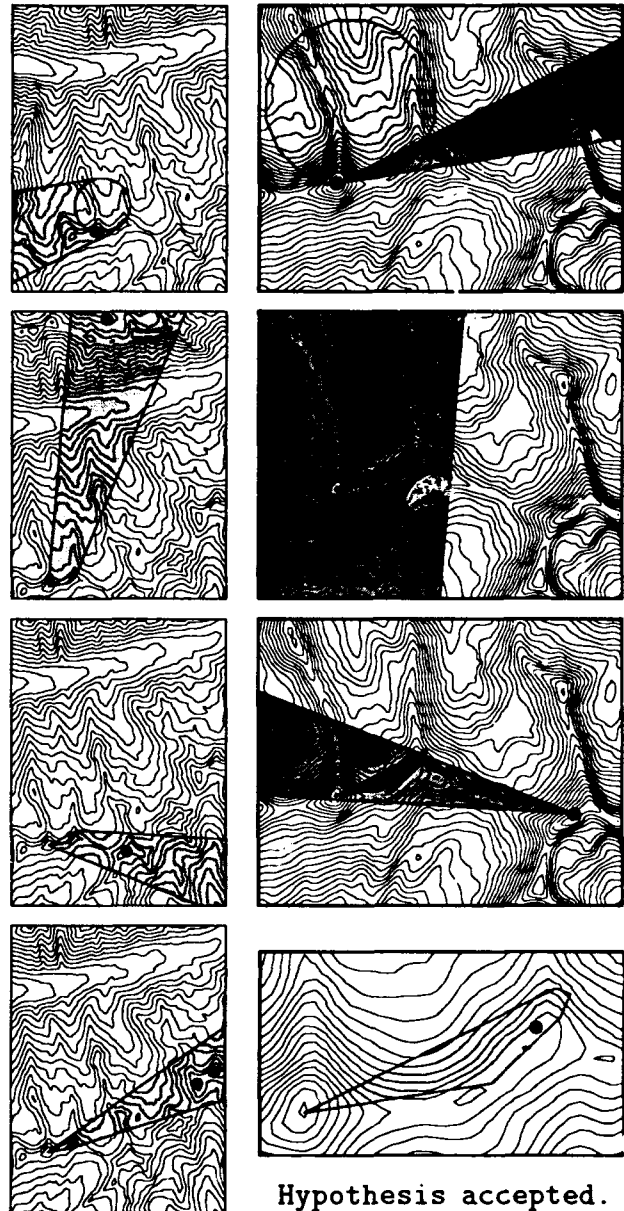


Figure 13: Viewpoint refinement to final region.

Figure 14 shows an attempt to refine one of the incorrect hypotheses. The upper left panel shows the map search region used to look for the most salient view peak. The feature being searched for is shown as an open circle. Due to the fact that the hypothesized viewpoint is wrong, a different feature was found, as indicated by the black dot. The viewpoint region was refined based on this match as shown to the right. The next most salient view feature was the long ridge line. As shown in the lower left panel, this was not found where expected and so the hypothesis was rejected. All five hypotheses not including the true viewpoint were rejected in a similar manner.

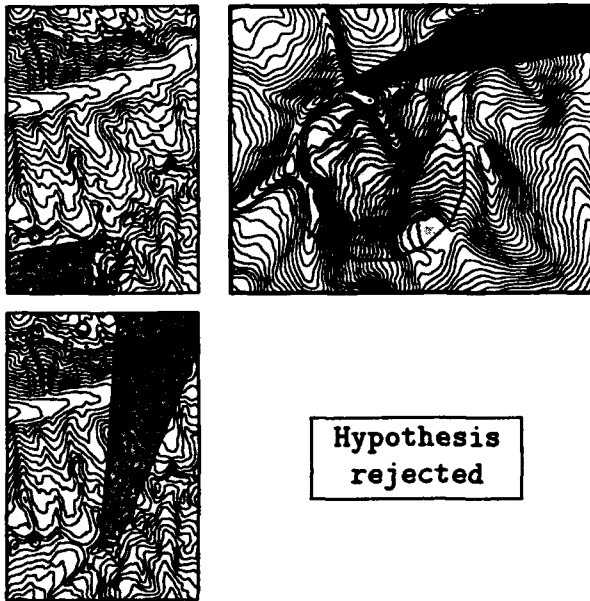


Figure 14: Validation of "incorrect" hypothesis.

Acknowledgements.

Rory Cejka provided many of the support tools used in this effort and assisted in the development of the edge thinning method. Sean Igo implemented the geometric reasoning subsystem.

References

- [Andreas et al., 1978] R.D. Andreas, L.D. Hostetler, and R.C. Beckmann. Continuous Kalman updating of an inertial navigation system using terrain measurements. In *Proc. IEEE National Aerospace Electronics Conference*, pages 1263-1270, 1978.
- [Baird and Abramson, 1984] C.A. Baird and M.R. Abramson. A comparison of several digital map-aided navigation techniques. In *Proc. IEEE Position Location and Navigation Symposium*, pages 294-300, 1984.
- [Ernst and Flinchbaugh, 1989] M.D. Ernst and B.E. Flinchbaugh. Image/map correspondence using curve matching. In *AAAI Symposium on Robot Navigation*, pages 15-18, March 1989.
- [Grimson, 1990] W.E.L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints*. The MIT Press, 1990.
- [Haralick et al., 1983] R.M. Haralick, L.T. Watson, and T.J. Laffey. The topographic primal sketch. *IEEE Journal of Robotics and Automation*, 2(1):50-72, 1983.
- [Heinrichs et al., 1992] M. R. Heinrichs, K. Smith, H. L. Pick, Jr., B. H. Bennett, and W.B. Thompson. Strategies for localization. In *Proc. DARPA Image Understanding Workshop*, January 1992.
- [Huttenlocher and Ullman, 1987] D.P. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proc. First International Conference on Computer Vision*, pages 102-111, 1987.
- [Levitt et al., 1987] T.S. Levitt, D.T. Lawton, D.M. Chelberg, and P.C. Nelson. Qualitative navigation. In *Proc. DARPA Image Understanding Workshop*, pages 447-465, February 1987.
- [Levitt et al., 1988] T.S. Levitt, D.T. Lawton, D.M. Chelberg, K.V. Koitzsch, and J.W. Dye. Qualitative navigation II. In *Proc. DARPA Image Understanding Workshop*, pages 319-326, April 1988.
- [Martelli, 1972] A. Martelli. Edge detection using heuristic search methods. *Computer Graphics and Image Processing*, 1(2):169-182, 1972.
- [Nevatia et al., 1992] R. Nevatia, K. Price, and G. Medioni. USC image understanding research: 1990-1991. In *Proc. DARPA Image Understanding Workshop*, pages 3-25, January 1992.
- [Pick et al., in press] H.L. Pick, Jr., M.R. Heinrichs, D.R. Montello, K. Smith, C.N. Sullivan, and W.B. Thompson. Topographic map reading. In J. Flach, P.A. Hancock, J.K. Caird, and K. Vicente, editors, *Ecology of Human-Machine Systems*. Lawrence Erlbaum Associates, (in press).
- [Sanson, 1973] F. Sanso. An exact solution of the roto-translation problem. *Photogrammetria*, 29:203-216, 1973.
- [Sha'ashua and Ullman, 1988] A. Sha'ashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *Second International Conference on Computer Vision*, pages 321-327, December 1988.
- [Stein and Medioni, 1992] F. Stein and G. Medioni. Map-based localization using the panoramic horizon. In *IEEE International Conference on Robotics and Automation*, Nice, France, May 1992.
- [Sutherland and Thompson, 1993] K.T. Sutherland and W.B. Thompson. Inexact navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 1993.
- [Talluri and Aggarwal, 1992] R. Talluri and J.K. Aggarwal. Position estimation for an autonomous mobile robot in an outdoor environment. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8:573-584, October 1992.
- [Thompson, 1958] E.H. Thompson. An exact linear solution of the problem of absolute orientation. *Photogrammetria*, 13(4):163-178, 1958.
- [Thompson, 1993] W.B. Thompson. Geometric constraints for viewpoint determination. University of Utah Computer Vision Group working paper, 1993.
- [Yacoob and Davis, 1991] Y. Yacoob and L.S. Davis. Computational ground and airborne localization over rough terrain. Technical Report CAR-TR-591, Computer Vision Laboratory, University of Maryland, 1991.

A simple, cheap, and robust visual navigation system

Ian Horswill

MIT Artificial Intelligence Laboratory
545 Technology Square
Cambridge, MA 02139, USA
ian@ai.mit.edu

Abstract

In this paper I will discuss a system which uses vision to guide a mobile robot through corridors and freespace channels. The system runs in an unmodified office environment in the presence of both static and dynamic obstacles (*e.g.* people). The system is among the simplest, most effective, and best tested systems for vision-based navigation to date. The performance of the system is dependent on an analysis of the special properties of robot's environment. I will describe these properties and discuss how they simplify the computational problems facing the robot.¹

1 Introduction

Navigation is one of the most basic problems in robotics. Since the ability to safely move about the world is a prerequisite of most other activities, navigation has received a great deal of attention in the AI, robotics, and computer vision communities. One of the limiting factors in the design of current navigation systems, as with many other robotic systems, has been the availability of reliable sensor data. Most systems have relied on the use of sonar data [7][8], or on vision [9][3][5][6][13][1]. In all cases, the unreliability of the available sensor data was a major concern in the research. Some researchers have even avoided the use of sensor data entirely in favor of precompiled maps [10].

In this paper, I will discuss a very simple vision-based corridor following system which is in day-to-day use here at MIT. The system is notable in that it is very fast (15 frames per second in the current system), very well tested, and uses only cheap "off-the-shelf" hardware. A major source of this simplicity is an analysis of the agent's niche. Such an analysis helps make clear the dependence of an agent on its environment and provides guidance for the design of future systems.

1.1 The Polly project

¹Support for this research was provided in part by the University Research Initiative under Office of Naval Research contract N00014-86-K-0685, and in part by the Advanced Research Projects Agency under Office of Naval Research contract N00014-85-K-0124.

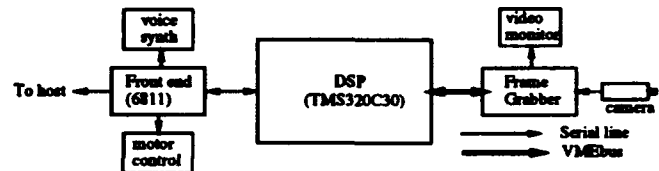


Figure 1: Hardware architecture of Polly.

Polly is a low cost, vision-based, autonomous robot built to help study how properties of the environment can simplify the computational problems facing an agent.² The theoretical goal of the project is to articulate a number of useful *computational properties* of office environments, and to develop a theory of how those properties can simplify the design of an agent. For example, one such property is that office environments do not generally have any moving objects other than people; thus, motion is a cue to agency. Rather than having to do a complicated analysis of the various objects in view to determine if they look or act like agents, the robot can simply check if they're moving, a much simpler test. This property does not hold of all habitats—trees blow in the wind and waves crash upon the shore, but they are not agents—and so the property partly determines of the set of habitats in which an agent that assumes it may survive. For this reason, I will refer to such properties as *habitat constraints* (see Horswill [4] for a more detailed discussion).

In this paper, I will discuss how habitat constraints can allow "optimizations" to be performed on the control systems of simple agents. The optimizations we will examine take the form of replacing a subsystem of the agent with another system which is in some way less expensive, but which nonetheless does the same job, provided that the habitat constraint holds. The motion constraint outlined above allows a motion detector to be substituted for a more complicated recognition system. This substitution is an optimization in the sense that the motion detector is less expensive than the recognition system. Depending on the context, we may measure expense as the actual monetary cost of building a

²Polly was built for \$20K (parts cost in the U.S.), but today, a roughly comparable, or perhaps even faster, system could be bought for roughly \$10K US.

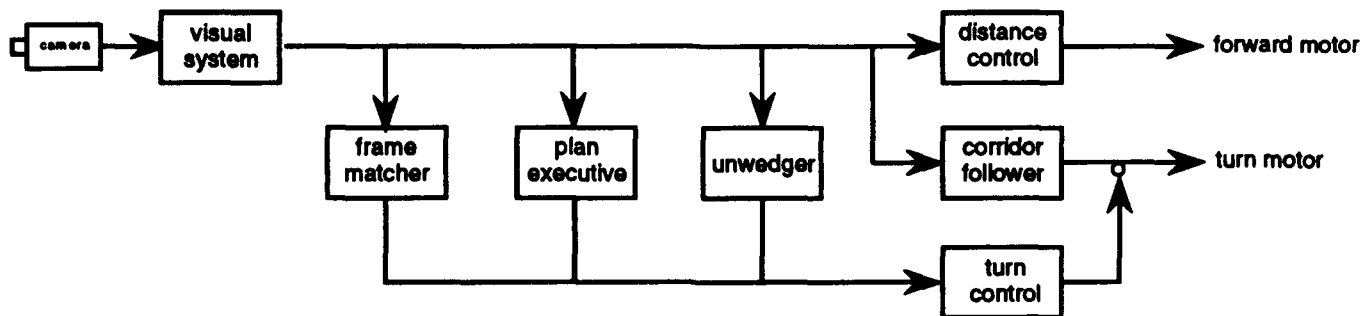


Figure 2: Conceptual architecture of the current version of the navigation system, as implemented in software.

robot, or the biological cost of growing and feeding extra neurons, or some other measure entirely. By analyzing an agent's specialization in terms of habitat constraints and optimizations, we can place specific properties of the environment into correspondence with specific computational problems facing the agent and the solutions which those properties allow. Such an analysis makes very explicit the way in which an agent is adapted to its environment, the possible consequences of changing that environment, and what facets of the agent would have to change to adapt to the new environment.

The implementation goal of the project is to use this approach to design to develop an efficient visual system which will allow the robot to run unattended for extended periods (hours) and to give primitive "tours" of the MIT AI lab. Our general approach to design has been to determine what particular pieces of information are needed by the agent to perform its activities, and then to design complete visual systems for extracting each piece of information. Thus, the agent might have distinct systems for answering questions such as "am I about to hit something?" and "what is the axis of this corridor?" If these systems are simple enough, they can be run in parallel very efficiently. In the system presented here, they are indeed simple enough so that each one can be run for each image frame, using an inexpensive computer.

The computational hardware on Polly consists of a 16MIP digital signal processor (Texas Instruments TMS320C30) with 64K 32-bit words of high speed ram³, a video frame buffer/grabber, a simple 8-bit microcontroller (M68HC11) for I/O tasks, and commercial microcontrollers for voice synthesis and motor control (see figure 1). Nearly all computation is done on the DSP. The fact that Polly uses only vision for sensing was due to lack of engineering time and experience, *not* because we feel that vision should be used for everything.

1.2 Work to date

At present, Polly consists of:

- A system for navigating corridors and relatively uncluttered spaces (reported on here)
- A rudimentary place recognition system

³The DSP includes an additional 1Mb of low speed ram, which is not presently in use. At present, only approximately 10KW of RAM are in use.

- A plan executive for forcing the robot to perform fixed sequences of actions (useful for debugging)
- A simple person detector based on bilateral symmetry
- An "unwedge," which pilots the robot out of cul-de-sacs and dead ends.
- A unit which overrides the corridor follower to perform open-loop turns
- A carpet-boundary detector (see section 7)

The connectivity of the navigation components is shown in figure 2. All these components are run in pseudo-parallel fashion on the DSP: at each moment in time, the robot grabs a new frame from the camera, runs each of the components, yielding a motor command, issues the motor command, and repeats the cycle.

The corridor follower nearly always controls the robot when it is moving. Even when the robot is not in a corridor, the corridor follower is still used to attempt to go forward without hitting obstacles. All other modules are built upon the corridor follower: the turn box can override the corridor follower to turn a corner; the place recognition system, the unwedger, and the plan executive can issue turn requests to the turn box. As of this writing, the corridor follower, unwedger, carpet boundary detector, and plan executive are essentially finished. The place recognition system and person detector are still under active development.

2 Corridor following

Corridor following is a common navigation task in office environments. Office buildings tend to consist of long corridors lined with rooms on either side, thus much of the work of getting from one room to another consists of driving along a series of corridors. The corridor follower described here is intended to be used as one component among many which cooperate to allow the robot to participate in its projects.

Corridor following can be broken into the complementary problems of keeping aligned with the axis of the corridor and keeping away from the walls. This amounts to keeping the variable θ in figure 3 small, while simultaneously keeping l and r comfortably large. Since Polly can only move in the direction in which it's pointed, these variables are coupled. In particular, if the speed of

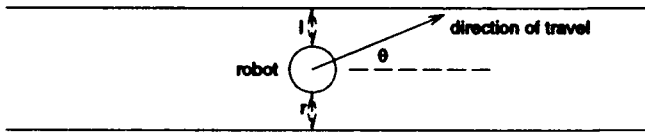


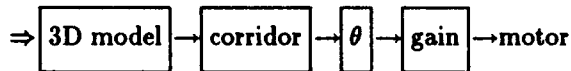
Figure 3: Corridor following.

the robot is s , then we have that:

$$\frac{dl}{dt} = -\frac{dr}{dt} = s \sin \theta$$

so moving away from a wall requires that Polly temporarily turn away from the axis of the corridor. Thus the problem for the control system amounts to controlling s and $\frac{d\theta}{dt}$ so as to keep l and r large and θ small and the problem for the visual system amounts to determining when one of these conditions is violated.

There is a huge space of possible solutions to this problem. Consider the subtask of aligning with the corridor. An obvious way of performing this task would be to use a system that first constructs a 3D model of the environment, then finds the walls of the corridor in the model, then computes the axis, θ , of the corridor from the walls, and finally, multiplies the axis by some gain to drive the turning motor. We can represent this schematically as:

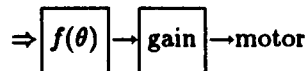


Here the double arrow at the beginning represents input from the sensors and the boxes are successive transformations of the sensory data. This is not a particularly efficient design however, since 3D models are both difficult and computationally expensive to build. Intuitively, building an entire model of the environment, only to compress down to a single number, θ , seems a waste of energy.

Of course, any system which computes θ , in whatever manner, and then turns to minimize θ , that is, any system of the form,



will do the trick. Furthermore, it can be shown that given the right conditions, we don't even need to compute θ *per se*; we can substitute any monotonic function of θ which is zero when θ is zero.⁴ Thus we can use any system of the form:



where f is the monotonic function.

The point of this analysis is simply that the constraints imposed by the task on the architecture of agent are actually quite modest, and that they admit a huge space of

⁴The conditions are that the motor must be controlled in velocity space, meaning that we have direct control over its speed, rather than just its acceleration, and that the control system must be fast enough so that we can treat it as having zero delay. If these conditions are not met, then the system may oscillate.

possible architectures. The choice of what architecture is best will depend on the resources available to the agent, the other tasks which the agent may have to perform, and the relative expense of the design.

3 The control system

At any given time, the corridor follower has to make a decision about what direction to turn, if at all, how fast to turn, and how fast to move forward. Since our robot has independent motors for turning and driving forward, we can treat these problems separately. In this section, I will discuss how the corridor follower computes the turn and drive rates from five numbers describing the situation: l' , r' , and θ' , which are estimates of l , r , and θ , respectively, c' , a measure of the distance to the nearest obstacle in front of the robot, and σ , a measure of the visual system's confidence in its estimate of θ . The control problem is actually much easier than the perceptual problem of estimating these numbers. In section 4, I will discuss two special properties of the environment which can make it easier for the visual system to estimate these numbers, and then in section 5, I will discuss the actual design of the visual system.

3.1 Steering for corridor following

the steering rate $\frac{d\theta}{dt}$ is controlled by θ' , σ , l' , and r' , which are measured by the visual system. The system drives the steering motor at a rotational velocity of

$$\frac{d\theta}{dt} = \alpha(l' - r') + \beta\theta'$$

if it is confident of its measure of θ' , otherwise with a velocity of $\alpha(l' - r')$. Here α and β are gains (constants) adjusted empirically for good results.

In practice, we have not measured the variable θ' directly, but instead have used the image plane x coordinate of the projection of the axis of the corridor, which is more easily measured. The projection x is equal to $k \tan^{-1} \theta$, where k is determined by the focal length and resolution of the camera. Thus the actual control law used in our system is:

$$\frac{d\theta}{dt} = \alpha(l' - r') + \beta \tan^{-1} \theta'$$

One could solve for θ given x and an accurately calibrated camera, but in our experience, this control system has worked perfectly well.

3.2 Controlling forward motion

The two constraints on forward velocity are that the robot should move forward when there's nothing in its way, and that it should stop when there is something close to it. We want the robot to move at full speed when the nearest obstacle is more than some safe distance, d_{safe} , and to stop when it's less than some distance d_{stop} . We use the rule

$$s = \min \left(v_{max}, \frac{v_{max}}{d_{safe} - d_{stop}} (c' - d_{stop}) \right)$$

which causes the robot to smoothly decelerate as it approaches an obstacle, and to back up if it gets too close to an obstacle. Backing up is useful because it allows one to herd the robot from place to place.

Constraint	Computational problem
Ground plane	Depth perception
Background-texture	Figure/ground separation
Long corridor edges	Vanishing point
Strong corridor edges	Edge detection (for vanishing point)
Known camera tilt	Vanishing point
Uniform non-corridor intersections	Vanishing point

Figure 4: Habitat constraints assumed by the visual system and the problems they helped to simplify. Note that "known camera tilt" is more a constraint on the agent, than on the habitat.

4 Computational properties of office environments

Estimating distance and parsing the visual world into objects are both very difficult problems in the general case. For example, figure/ground separation can be arbitrarily difficult if we consider pathological situations such as camouflage or crypsis, which can require predators to learn to recognize prey on a case by case basis (see Roitblat [12], p. 260). Fortunately, office environments, Polly's habitat, are actively structured by both designers and inhabitants so as to facilitate their legibility (see Passini [11] for an interdisciplinary discussion of the navigational properties of buildings). The special properties of office environments allow much simpler architectures to be used than would be necessary for an agent which was to follow arbitrary paths in arbitrary environments.

One such property is that office environments have a flat ground plane, the floor, upon which most objects rest. For a given height and orientation of the camera, the distance of a point P from the camera will be a strictly increasing function of the height of P 's projection in the image plane, and so image plane height can be used as a measure of distance to objects resting on the floor⁵. While this is not a linear measure, and the exact correspondence between heights and distances cannot be known without first knowing the specifics of camera, it is still a perfectly useful measure for determining which of two objects is closer, whether an object is closer than a certain threshold, or even as an (uncalibrated) measure of absolute distance. This property was referred to as the *ground plane constraint* in [4].

Another important property of office environments is that they are generally carpeted, and their carpets generally have only fine-scale texture. That is to say, that from a distance, the carpet will appear to have a uniform reflectance. If this is true, and if the carpet is uniformly illuminated, then the areas of the image which correspond to the floor should have uniform image brightness, and so any violation of this uniformity must be an object other than the floor. This property, called the *background texture constraint* in [4], can greatly simplify the computational problem of figure-ground separation.

5 Design of the visual system

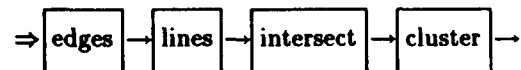
The visual system estimates the axis of the corridor and three distance measures from 64×48 pixel grey-scale im-

⁵This observation goes back at least to Euclid. See [2].

ages covering a field of view of 110 degrees (1.9 radians). All vision computations are performed for each frame. The overall structure of the visual system is shown in figure 5. The constraints used in the optimization of the system are given in figure 4.

5.1 Computing the vanishing point

As was mentioned above, the axis of the corridor is represented by the x coordinate of its image-plane projection. This can be estimated by finding the vanishing point of the parallel lines forming the edges of the corridor. Bellutta *et al* [1] report on a such a system which extracts vanishing points by running an edge finder, extracting straight line segments, and performing 2D clustering on the pairwise intersections of the edge segments.⁶ We can represent this schematically as:

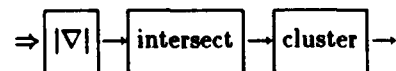


This algorithm, while less computationally expensive than 3D modeling, is still rather expensive. We can simplify the system if we make stronger assumptions about the environment. We can remove the step of grouping edge pixels into segments by treating each edge pixel as its own tiny segment:



Note that this system will effectively weight segments by their length. This will be fine provided that the long lines in the scene are the lines directed toward the vanishing point.

Edge detectors can also be extremely expensive. Since the edges we're looking for should be very strong and straight, we should be able to use a very simple edge detector, such as a gradient threshold. Computing the intensity gradient (the rate of change in image brightness, denoted by ∇) at a pixel and testing its magnitude can be done using only a few machine instructions. The resulting system is then:



If the tilt-angle of the camera is held constant by the camera mount, then the vanishing point will always have

⁶The algorithm of Bellutta *et al.* is actually more complicated than this in that it extracts multiple vanishing points, but for our purposes we can treat it as extracting only the forward vanishing point.

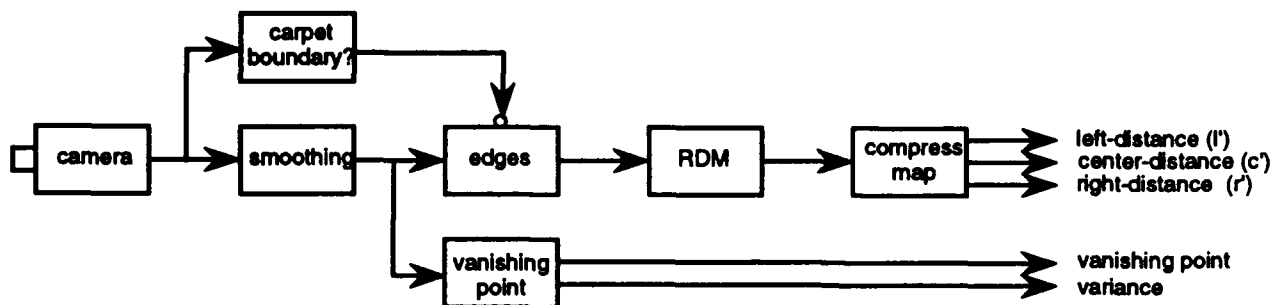
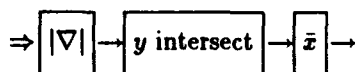


Figure 5: The portion of the visual system devoted to corridor following. Note that smoothing is performed prior to edge detection to remove noise. The vanishing point unit is shown as a single box because all the steps of the vanishing computation are performed simultaneously. The carpet boundary detector is discussed in section 7.

the same y coordinate, that is, it will lie in the line $y = y_0$ for some y_0 . We can then replace the pairwise intersections with the intersections of the edges with $y = y_0$. This reduces the number of points to consider from $O(n^2)$ to $O(n)$, where n is the number of edge pixels, and also reduces the clustering to a 1D problem, which is also more efficient:



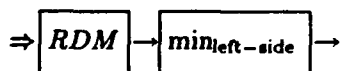
Finally, if we assume that the intersections of the edges not pointing toward the vanishing point are uniformly distributed, then we can replace the clustering operation, which looks for modes, with the mean of the x coordinate:



This does have the disadvantage that if there are many non-corridor edges in view, then the mean will tend to move toward the center of the screen. The sign will still be correct however, and so the robot will still steer in the correct direction.

5.2 Computing distances

There are two problems involved in estimating the left, right, and center distances: figure/ground separation to find the walls in the image, and depth estimation to determine the distances to them. Suppose we had computed a radial depth map, RDM , of the scene. A radial depth map gives the distance of the nearest object in each direction. Then we could find the distance to the left wall by finding the minimum distance given in the left side entries of the radial depth map:

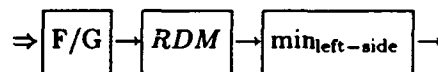


If we had already somehow solved the figure-ground problem, that is, if we had already labeled every pixel as being either "floor" or "not floor", then we could use the ground-plane constraint to generate the RDM : the distance to the closest object in a given direction is a monotonic function of the height in the image plane of the lowest non-floor pixel in the image plane. Since columns of the image correspond to directions, the distance is

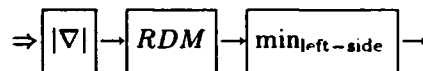
then simply the height of the lowest non-floor pixel in a given column. Thus:

$$RDM(x) = \min\{y | \text{the point } (x, y) \text{ isn't floor}\}$$

so now our system looks like this:



where "F/G" is the figure/ground system. If the floor is textureless but the walls generate edges where they meet with the floor, then, by the background-texture constraint, it too can be replaced, in this case, by an edge detector (e.g. thresholded gradients):



A number of things are worth noting here. First of all, l' and r' are not necessarily the distances to the walls. They are simply the distances to the nearest non-floor objects on the left and right sides of the image. This is not actually a problem however, since if there are other objects in the way it will simply cause the robot to steer around them, thus conferring on the robot a limited object avoidance capability. If there are no such objects, then there is no difference anyhow. Thus having the system make no distinctions between walls and other obstacles is actually advantageous in this situation. The second thing worth noting is that the distance measures are nonlinear functions of the actual distances⁷. For some applications this might be unacceptable, but for this application we are mostly just concerned with whether a given object is too close or whether the left side or the right side is closer, for which these nonlinear measures are quite adequate. Finally, since no camera has a 180 degree field of view, l' and r' are not even measures of the perpendicular distances to the walls, l and r , but rather are measure of the distance to the closest point in view. Again, this is not a problem in practice, partly because our camera has a relatively wide field of view, and partly because for a given orientation of the robot, the perpendicular distance is another monotonic and strictly increasing function of the measured distance, and *vice versa*.

⁷The actual function is a quotient of linear equations whose coefficients are determined by the camera parameters.

Test	Time (sec)	Frames/sec
Full system	67	15
Corridor follower	67	15
I/O only	67	15
No I/O	15	67
No VP	10	100

Figure 6: Execution times for 1000 frames. "Full system" is all code presently implemented, including the person detector. "No I/O" is the corridor follower without any frame grabbing or output to the base (a single frame is grabbed at the beginning and processed repeatedly). "No VP" is the collision avoidance system run without I/O or the vanishing-point box. All execution times are for a Texas Instruments TMS-320C30-based DSP board (a Pentek 4283) running with no wait states. The processor has a 60ns instruction time. The first three lines are the same because the system cannot digitize frames faster than 15 FPS.

6 Evaluation

The corridor follower has been running for nine months and is quite stable. It has seen at least 200 hours of service with many continuous runs of one hour or more. This makes it one of the most extensively tested and reliable visual navigation systems to date. We have been able to run the system as fast as our robot base could run without shaking itself apart (approximately 1 m/s). While there are cases which will fool the braking system (see below), we have found the system to be quite reliable in general.

6.1 Efficiency

The system is very efficient computationally. The present implementation runs at 15 frames per second, which is as fast as the system can read data from the camera (see figure 6). All computation boxes in figure 5 are run on every frame in (simulated) parallel. This implementation is heavily I/O bound however, and so it spends much of its time waiting for the serial port and doing transfers over the VMEbus to the frame grabber and display. We expect that performance would be noticeably better on a system with a more tightly coupled DSP and frame grabber. This efficiency of the system allows it to be implemented with a relatively simple and inexpensive computer such as the DSP. The modest power requirements of the computer allow the entire computer system to run off of a single motorcycle battery for as long as nine hours.

The simplicity and efficiency of the system make it quite inexpensive compared to other real-time vision systems. C30 DSP boards are now available for personal computers for approximately \$1-2K US and frame grabbers can be obtained for as little as \$400 US. Thus the corridor follower would be quite cheap to install on an existing system. We are also working on a very inexpensive hardware platform for the system which we hope will cost less than \$200 US.

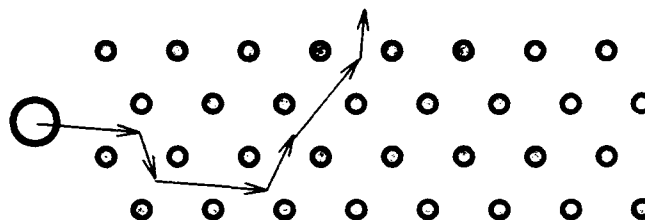


Figure 7: A forest environment.

6.2 Failure modes

The system runs on all floors of the AI lab building on which it has been tested (floors 3-9) except for the 9th floor, which has very shiny floors. There the system brakes for the reflections of the overhead lights in the floor. The present system also has no memory and so cannot brake for an object unless it is actually in its field of view. This sometimes causes problems. The system also cannot brake for an object unless it can detect an edge on or around it, but this can more or less be expected of all vision systems. The system's major failure mode is braking for shadows. If shadows are sufficiently strong they will cause the robot to brake when there is in fact no obstacle. This is less of a problem than one would expect because shadows are generally quite diffuse and so will not necessarily trigger the edge detector. Finally, the 7th floor of the lab, where the robot spends most of its time, does not have a single carpet, but several carpets, each with a different color. The boundaries between these carpets can thus be mistaken for obstacles. This problem was dealt with by explicitly recognizing the boundary (see section 7).

6.3 Performance outside corridors

While the system was designed to navigate corridors, it is also capable of moving through more complicated spaces. Its major deficiency in this situation is that there is no way of specifying a desired destination to the system. Effectively, the system acts in a "point and shoot" mode: it moves forward as far as possible, veering away from obstacles, and continuing until it reaches a dead end or is stopped externally. The system is also non-deterministic in these situations. When the robot is blocked by an object, it will turn either left or right depending on the exact position and orientation of the robot and object. Since these are never exactly repeatable, the robot is effectively non-deterministic. Thus in a "forest environment" such as figure 7, the robot could emerge at any point or even get turned around completely. The system's performance is good enough however that a higher-level system can lead it through a series of corridors and junctions by forcing the robot to make a small open-loop turn when the higher-level system wants to take a new corridor at a junction. The corridor follower then realigns with the new corridor and continues on its way.

7 Extensions

A number of minor modifications to the algorithm described above are worthwhile.

Vertical biasing

As discussed above, shadows and bright lights radiating from office doors can sometimes be sufficiently intense to trigger the edge detector. Since these shadows always radiate perpendicular to the wall, they appear horizontal in the image when the robot is successfully aligned with the corridor. By biasing the edge detector toward vertical lines, we can make it less sensitive to these shadows. A previous version of the system dealt with the problem by weighting vertical lines twice as much as horizontal lines. The system now explicitly searches for carpet boundaries and briefly disables the detection of horizontal lines when a carpet boundary is found. The criterion for a carpet boundary is that it must be a weak horizontal line with no surrounding texture.

Fear of the dark

The system, as described above, will happily drive through a dark room and hit the nearest obstacle. Similarly, if the robot somehow misses the boundary between the floor and a blank wall, and drives close enough to the wall for it to fill the robot's visual field, then it will treat the blank wall as being an empty floor and attempt to drive up the wall. Polly avoids these problems by treating dark pixels as obstacles to be avoided, and by refusing to drive forward if there is insufficient texture in the image. A better solution would be to add other sensory modalities to the system, such as touch or sonar, but those sensors were not available on the robot.

Wall following

When the system described above reaches a large open space, the single wall which is in view will act as a repulsive force, causing the robot to turn away from it until there is nothing in view whatsoever. Thus it naturally tends toward situations in which it is effectively blind. While this is sufficient for following corridors, and is in fact a very good way of avoiding obstacles, it is a very bad way of actually getting to a destination unless the world consists only of nice straight corridors. This problem can be fixed by modifying the steering control so that when only a single wall is in view, the robot will try to keep that wall at a constant distance. Thus, in the case where only the left wall is in view, the control law becomes:

$$\frac{d\theta}{dt} = \alpha(l' - d)$$

Where d is the desired distance to the wall.

8 Conclusions

Curiously, the most significant things about the system are the things which it does not do. It does not build or use detailed models of its environment; it does not use carefully calibrated depth data; it does not use high resolution imagery; and it is not designed to run in arbitrary environments. Indeed, much of its power comes from its specialization.

One may be tempted to object that this system is too domain specific and that more complicated techniques are necessary to build practical systems for use in the real world. I think that this is misguided however. To begin with, even if one had a single truly general navigation algorithm, its very generality would likely make it much slower than the system discussed here. The general system may also require allocating scarce cognitive or attentive resources which would be better used for other concurrent tasks. One approach would be to build a hybrid which used the simple system when possible, and the more cumbersome system only when necessary.

Another possibility would be to build a system which could rapidly switch between a number of domain-specific strategies. Ullman's Visual Routine Processor [14] is a particularly attractive architecture for this approach. A VRP could be quickly configured by the central system to use different strategies for different situations. Ideally, such a system would be able to recognize and learn to use domain-specific strategies for visual tasks, thus making it truly adaptive (see Whitehead and Ballard for an interesting example of learning visual routines [15]).

It remains to be seen how far we can go with simple, domain-specific strategies which rely on special properties of the environment. I suspect that quite a lot can be done with them. In either case, the system described here is a demonstration that it is practical to build simple, inexpensive vision systems which perform useful tasks, and that the solutions to vision problems do not necessarily involve buying better cameras and bigger computers.

Acknowledgements

Both this paper and the system described within benefited from discussions with Lynn Stein, Rod Brooks, Eric Grimson, Maja Mataric, Mike Bolotski, Jose Robles, and David Michael.

References

- [1] P. Bellutia, G. Collini, A. Verri, and V. Torre. Navigation by tracking vanishing points. In *AAAI Spring Symposium on Robot Navigation*, pages 6-10, Stanford University, March 1989. AAAI.
- [2] James E. Cutting. *Perception with an Eye for Motion*. MIT Press, 1986.
- [3] Y. Goto and A. Stenz. The cmu system for mobile robot navigation. In *1987 IEEE International Conference on Robotics and Automation*, pages 99-105. IEEE, March 87.

- [4] Ian Horswill. Characterizing adaptation by constraint. In *Proceedings of the First Annual European Conference on Artificial Life*, 1991.
- [5] Ian D. Horswill. Reactive navigation for mobile robots. Master's thesis, Massachusetts Institute of Technology, June 1988.
- [6] A. Kosaka and A. C. Kak. Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties. *Computer Vision, Graphics, and Image Processing*, 56(3), September 1992.
- [7] Maja Mataric. A distributed model for mobile robot environment-learning and navigation. Technical Report 1228, Massachusetts Institute of Technology, Artificial Intelligence Lab, May 1990.
- [8] Hans P. Moravec. Certainty grids for mobile robots. Unpublished memo.
- [9] Hans P. Moravec. The stanford cart and cmu rover. Technical report, Robotics Institute, Carnegie-Mellon University, February 1983.
- [10] ed. Nils J. Nilsson. Shakey the robot. Technical Report 323, SRI International, April 1984.
- [11] Romedi Passini. *Wayfinding in Architecture*, volume 4 of *Environmental Design Series*. Van Nostrand Reinhold, New York, 1984.
- [12] Herbert L. Roitblat. *Introduction to Comparative Cognition*. W. H. Freeman and Company, 1987.
- [13] K. Storjohann, T. Zeilke, H. A. Mallot, and W. von Seelen. Visual obstacle detection for automatically guided vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 761-766, May 1990.
- [14] Shimon Ullman. Visual routines. *Cognition*, 18:97-159, 1984.
- [15] S. Whitehead and D. Ballard. Active perception and reinforcement learning. *Neural Computation*, 2(4), 1990.

Model Extension and Refinement using Landmarks

Harpreet S. Sawhney Rakesh Kumar Allen R. Hanson

Computer Science Department

University of Massachusetts

Amherst, MA 01003

Phone : (413)545-2744

ABSTRACT

Least-squares and robust methods for determining outliers have been effective in the location and orientation of a mobile robot from visual measurements of modeled 3D landmarks. However, building the initial 3D landmark models is a time consuming and tedious process. For landmark-based navigation methods to be widely applicable, automatic methods have to be developed to build new 3D models and enhance the existing models. Ideally, a robot would continuously build and update its world model as it explores the environment. This paper presents techniques to determine the 3D location of image features from a sequence of monocular 2D images captured by a camera mounted on the robot.

The approach adopted here is to first build a partial model (possibly noisy) by tracking and reconstructing *shallow* structures over a sequence of images using the constraint of affine trackability. This model is subsequently used to compute the pose that relates the model coordinate system and the camera coordinate system of the image frames in the sequence. Unmodeled 3D features are then tracked over the image sequence and their 3D locations recovered by a pseudo-triangulation process, a form of "induced stereo". The triangulation process is also used to make new 3D measurements of the initial model points. These measurements are then fused with the previous estimates to refine the set of initial model points.¹

1 Introduction

Techniques are presented for initial model acquisition, and then model extension using a partial model to relate the camera and world/model coordinate systems. The partial model is derived from the reconstruction of shallow² environmental structure [Sawhney and Han-

son, 1992a]. Model extension results are also presented for one sequence where the partial model was manually built.

1.1 Related Work

Previous research on multi-frame 3D reconstruction can be categorized into two broad classes. The first class assumes that a model of 3D inter-frame motion is known, rather than assuming independent motion parameters between consecutive frames. Broida [Broida and Chelappa, 1991] assumes constant velocity motion and estimates the 3D location of a set of points tracked over a monocular image sequence. [Chandrasekhar, 1991] extended Broida's technique to deal with data sets where the 3D location of a few points is known. The objective function, which Broida and Chandrasekhar et. al. minimise, has the motion model parameters and the unknown structure location parameters as unknowns. Thus the dimension of the objective function grows with the number of unknown points. An even more basic limitation of this approach lies in the model of motion being adopted and its suitability to the motion being observed.

The second class of techniques does not assume any model of motion. The rigid structure of the world is carried forward by the depth estimates from frame to frame. These techniques are sequential in nature and typically use Kalman Filtering to compute the depth estimates [Cui, et al, 1990], [Oliensis, 1991], [Sawhney, 1991]. Oliensis and Thomas [Oliensis, 1991] solve for the motion parameters between consecutive image frames in a monocular image sequence. With each image pair, new measurements are made for depth values of features and these are integrated with previous estimates in the Kalman Filter framework. The new observation Oliensis and Thomas [Oliensis, 1991] make is that the depth estimate of different feature points are correlated since the same noisy motion parameters are used to compute the depth. Because of this correlation, they estimate the depth parameters of all points simultaneously. This gives them fairly good depth estimates for camera motions having some T_z (i.e. translation along the optical axis) component. The cost, however, is that for estimat-

¹This work was supported in part by DARPA (via TACOM) under contract number DAAE07-91-C-R035, and by NSF under grant number CDA-9822572.

²Shallow structures have small extent in depth compared to their distance from the camera.

ing the depths of m points, a covariance matrix of size $3m \times 3m$ must be inverted with each new frame.

1.2 Overview

All of these approaches rely on the basic principle of triangulation to reconstruct new 3D points. However, reconstruction by triangulation is highly sensitive to errors in estimating the relative orientation between consecutive camera frames. In this paper, the reconstruction of 3D structure is accomplished in two steps to overcome this limitation.

The first step is a partial reconstruction of a scene in terms of *shallow* 3D environmental structure; structures whose extent in depth is small compared to their distance from the camera. The 3D motion and structure of a shallow object in motion, relative to the camera, can be well approximated by an affine transformation. In [Sawhney, 1991], a framework was presented for tracking shallow objects over time under the affine constraint, and in [Sawhney, 1992b] an algorithm for identification and 3D reconstruction of these structures is presented. An important advantage of this approach is that 3D structure is derived reliably without the intermediate step of explicit computation of the 3D motion parameters.

Shallow structure reconstruction provides only a partial 3D model for the scene. However, this partial model is adequate for the second part of the technique presented in this work, namely model extension and refinement. The partial model is used to compute the pose that relates the model coordinate system and the camera coordinate system of the image frames in the sequence. The unmodeled 3D features (those not recovered by the shallow structure reconstruction) are tracked over the image sequence using an optic-flow-based line tracking algorithm [Williams, 1989]. Using correspondences of image features, and the poses computed from model-to-image feature correspondences for a sequence, new 3D points are located by triangulation (see Figure 1). The estimation of the new 3D points is done using both batch and quasi-batch or sequential methods. The triangulation process is also used to make new 3D measurements of the initial model points, which are then fused with the previous estimates to refine the set of initial model points. Results are presented for real sequences where new 3D points are located with average errors of 1.76 %.

2 Shallow Structure Reconstruction

This section presents a brief summary of identification, tracking and 3D reconstruction of shallow structures. The details can be found in [Sawhney, 1992a].

Given a 3D structure that can be well approximated by a fronto-parallel plane (shallow structure), its image projections at two closely spaced time instants are related

through:

$$\frac{1}{f}p' \approx \frac{1}{f}sR_s p + t, \quad t = s\Omega_{xy} + \frac{1}{Z'_0}T_{xy} \quad (1)$$

where, p and p' are the corresponding imaged points of a shallow structure at times n and $n+1$ respectively, s is the scale defined as the ratio of average depths at the two time instants, R_s is the 2×2 rotation matrix for the rotation around the optical axis (z -axis), t is the translation in the image plane, Ω_{xy} and T_{xy} are the vectors representing the x and y components of the 3D rotational and translational vectors respectively, Z'_0 is the average depth at the second time instant, and f is the focal length of the camera.

A set of noisy line correspondences are used to compute the best affine motion parameters in the image plane. An error measure that is a weighted sum of the parallel and perpendicular components of the vectors joining the corresponding endpoints of the line in frame $n+1$ and the affine transformed line in frame n is formulated:

$$E_i = \sum_{j=1}^2 w_{\perp i} [(D_{ij}r_s + t - p'_{ij}) \cdot n'_i]^2 + w_{\parallel i} [(D_{ij}r_s + t - p'_{ij}) \cdot l'_i]^2 \quad (2)$$

where i is the i th corresponding pair, j refers to endpoint 1 or 2, $w_{\perp i}$ and $w_{\parallel i}$ are the weights for the perpendicular and parallel error components, $D = \begin{bmatrix} x & -y \\ y & x \end{bmatrix}$ is the data matrix which is constructed using the endpoint $p = [x \ y]^T$ in frame n , vector $r_s = [s \cos \omega_s \ s \sin \omega_s]^T$ is the product of scale s and rotation, ω_s , around the optical axis, and n'_i and l'_i are the unit normal and direction, respectively, of the line in frame $n+1$.

For a set of line correspondences, the unknown parameters r_s and t can be found by minimising $\sum_i E_i$ which leads to a linear system:

$$M_{tot} v_{aff} = v_{tot} \quad (3)$$

where M_{tot} and v_{tot} are the data matrix and vector, respectively, and v_{aff} is the vector of the unknown four affine parameters (for full details, see [Sawhney, 1992a]).

Given the model of uncertainty of the constituent lines in a structure, the covariances of the output affine parameters can be expressed as follows [Strang, 1986]:

$$\Lambda_{r,t} = M_{tot}^{-1} \quad (4)$$

where $\Lambda_{r,t}$ is the 4×4 covariance matrix of the affine parameters r_s and t .

2.1 Tracking Shallow Structures

The affine motion constraint is used in a dynamic model to predict and track shallow structures over time. Tracking requires:

1. A dynamic model of the motion of a structure.

2. A match measure to choose good matches for a structure in every newly acquired frame. The constraints on the search for the potential matches are provided by the dynamic model.
3. A mechanism for fusing the current estimate of the affine motion and the 3D location parameters of a structure with those obtained from the newly acquired data.

The affine motion parameters derived in Equation 3 provide a dynamic model of prediction of the motion of a shallow structure in the image plane. Kalman filtering is used for prediction and recursive estimation, and the Mahalanobis distance [Mahalanobis, 1936] is used for matching the predictions with potential matches in a newly acquired frame [Sawhney, 1991].

2.2 Shallow Structure Identification and Reconstruction

In [Sawhney, 1992b] affine tracking is embedded in an algorithm to automatically identify shallow structures. The essential idea is that if a hypothesised structure can be consistently tracked and its 3D depth over time is consistent with a shallow structure model, then the structure is identified as shallow; otherwise it is labeled non-shallow. The depth of structures identified as shallow is computed from the scale parameter in the affine transformation of Equation 3. It is represented in the coordinate system of the first frame in the sequence.

3 Pose Determination

Using the depths of the shallow structures recovered by the affine-based algorithm, a partial model of the environment can be built. This model has the same coordinate system as that of the first frame's coordinate system. Given correspondences between model and image tokens in subsequent image frames, the pose parameters (rotation and translation) that relate the subsequent frames' coordinate systems to the model coordinate system can be computed. In an earlier paper [Kumar, 1989] least-squares techniques for pose determination were developed. These techniques are optimal with respect to gaussian noise in the input image measurements. In this section, the least-squares techniques are extended to also handle gaussian noise in the 3D model. The techniques presented in this section assume point correspondences but are easily modified for line correspondences.

The rigid body transformation from the world coordinate system to the camera coordinate system can be represented as a rotation (R) followed by a translation (\vec{T}). A point \vec{p} in world coordinates gets mapped to the point \vec{p}_c in camera coordinates as:

$$\vec{p}_c = R(\vec{p}) + \vec{T} \quad (5)$$

Using equation (5) and assuming perspective projection, the pose constraint equations for the i th point \vec{p}_i in a set of " m " points can be written in the following manner:

$$\frac{1}{p_{csi}} \vec{C}_{si} \cdot (R\vec{p}_i + \vec{T}) = 0 \quad (6)$$

$$\frac{1}{p_{csi}} \vec{C}_{yi} \cdot (R\vec{p}_i + \vec{T}) = 0 \quad (7)$$

$$\vec{C}_{si} = (s_x, 0, -I_{si}) \quad (8)$$

$$\vec{C}_{yi} = (0, s_y, -I_{yi}) \quad (9)$$

$$p_{csi} = (R\vec{p}_i + \vec{T})_z \quad (10)$$

where (I_{si}, I_{yi}) is the image projection of the point and (s_x, s_y) is the focal length in pixels along each axis.

The non-linear system of constraint equations for the pose parameters R and \vec{T} is solved using the gauss-newton technique [Strang, 1986]. Given a current estimate R, \vec{T} , the constraint equations (6,7) are linearised about the estimate:

$$\frac{1}{p_{csi}} (\vec{C}_{si} \cdot \Delta \vec{T} + \delta \vec{\omega} \cdot \vec{b}_{si}) = -\frac{1}{p_{csi}} \vec{C}_{si} \cdot \vec{p}_{ci} + \eta_x \quad (11)$$

$$\frac{1}{p_{csi}} (\vec{C}_{yi} \cdot \Delta \vec{T} + \delta \vec{\omega} \cdot \vec{b}_{yi}) = -\frac{1}{p_{csi}} \vec{C}_{yi} \cdot \vec{p}_{ci} + \eta_y \quad (12)$$

where $\vec{b}_{si} = R\vec{p}_i \times \vec{C}_{si}$ and $\vec{b}_{yi} = R\vec{p}_i \times \vec{C}_{yi}$. The above equations relate the pose increments $\delta \vec{\omega}$ (rotation) and $\Delta \vec{T}$ (translation) to the computed measurement errors using the current pose estimate. The noise terms in the two equations, η_x and η_y are functions of both the 3D model noise $\Delta \vec{p}_i$ and the image noise $\Delta X, \Delta Y$:

$$\eta_x = \Delta X + \frac{1}{p_{csi}} \vec{C}_{si} \cdot (R(\Delta \vec{p}_i)) \quad (13)$$

$$\eta_y = \Delta Y + \frac{1}{p_{csi}} \vec{C}_{yi} \cdot (R(\Delta \vec{p}_i)) \quad (14)$$

Therefore for the i th point, two such equations (11 and 12) can be written and for a set of " m " points, a total of " $2m$ " equations are obtained. At each iteration in the minimisation process, the linear system of equations is solved using equation (23) to find the best increment vector³. This increment is added to the current pose estimate and the process repeated until there is convergence.

If the correct estimate of pose were known, the measurement noise terms η_x and η_y would be equal to the sum of the measurement error of the image point location and the projection of the error in the model point along the image x-axis and y-axis respectively. The measurements of the image point locations are assumed to be corrupted with identical, independent, zero-mean gaussian noise. The 3D model points are also assumed to be corrupted

³The appendix reviews some salient information on solving over constrained linear equations.

by zero-mean independent gaussian noise. Thus the covariance matrix "V" corresponding to the noise in the linear system of equations (22) in the Appendix is a band matrix in which the non-zero entries are 2×2 matrices about the diagonal. The output covariance matrix for the pose rotation and translation parameters is given by equation (24) evaluated at the final pose estimate.

4 Induced Stereo

In this section, we present techniques for computing 3D estimates of new points in the world coordinate system from their tracked image locations over a multi-frame sequence. The mathematics for both extending the model and refining the initial modeled points is presented. Computed with the estimate of each new model point is an estimate of the covariance of its error. These covariances are functions of the input image measurement covariances and the initial 3D model point covariances.

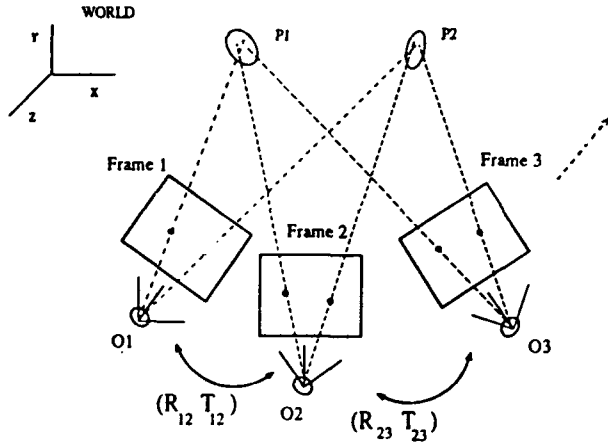


Figure 1: Model Extension and Refinement.

The matching of image features to the partial model is obtained by the tracking method described in Section 2. Given these correspondences, pose estimation is performed for each frame using the method presented in the previous section. Image tokens corresponding to new features are also tracked over a sequence of frames using the computed optic flow between pairs of successive frames [Williams, 1989]. Typically corner points (defined by the intersection of two image lines) are tracked although any image feature which can be reliably tracked may be used. The 3D estimate of the corner point is obtained by the pseudo-intersection of all the image projection rays for a tracked image point. A nice property of the system described here is that the pose estimation process provides the world coordinate frame as a stable coordinate frame in which 3D measurements from a sequence of frames can be combined. Independent measurements can be

made to relate the coordinate system of each frame in the sequence to the world coordinate frame.

Points are located by the pseudo-intersection process in two steps. In the first step, a 3D error function is minimised to find an initial estimate of the point's location. This step, however, does not yield the optimal estimate since the various error terms are not weighted by the input covariances. In the second step, an image-based error function is optimised in which the error terms are inversely weighted by a combination of the input covariances of the pose estimate and the image measurements.

Let r_i be the unit vector corresponding to the image projection ray for an image point in the i th frame. The pose estimation for this frame is given by the rotation R_i and translation \vec{T}_i . Since the image projection rays do not intersect at a unique point⁴, the 3D pseudo-intersection point \vec{p} is obtained by minimising an error function E :

$$E = \sum_{i=1}^n \|(R_i(\vec{p}) + \vec{T}_i) \times r_i\|^2 \quad (15)$$

which is the sum of squares of the perpendicular distances from the pseudo-intersection point \vec{p} to the image projection rays. Differentiating E with respect to the unknown variable \vec{p} leads to a set of linear equations, which are then solved to give the initial estimate for \vec{p} .

In the second step, the pose constraint equations 6 and 7 are used to formulate image-based error equations for the X and Y projections of the model points.

$$\frac{1}{p_{cs}} \vec{C}_{xi} \cdot R_i(\vec{p}) = -\frac{1}{p_{cs}} \vec{C}_{xi} \cdot \vec{T}_i + \zeta_X \quad (16)$$

$$\frac{1}{p_{cs}} \vec{C}_{yi} \cdot R_i(\vec{p}) = -\frac{1}{p_{cs}} \vec{C}_{yi} \cdot \vec{T}_i + \zeta_Y \quad (17)$$

where ζ_X and ζ_Y are the noise terms that are functions of both noise in pose $\Delta \vec{T}_i$ and $\delta \vec{\omega}_i$ and image noise $(\Delta X, \Delta Y)$:

$$\zeta_X = \Delta X + \frac{1}{p_{cs}} \vec{C}_{xi} \cdot \Delta \vec{T}_i + \frac{1}{p_{cs}} \delta \vec{\omega}_i \cdot \vec{b}_i \quad (18)$$

$$\zeta_Y = \Delta Y + \frac{1}{p_{cs}} \vec{C}_{yi} \cdot \Delta \vec{T}_i + \frac{1}{p_{cs}} \delta \vec{\omega}_i \cdot \vec{b}_i \quad (19)$$

In this case the 3D model point \vec{p} is the unknown variable. The denominator p_{cs} in the equations (16 and 17) corresponds to the depth of the point and is a function of the unknown variable \vec{p} . Therefore, for each frame over which the point is tracked, two non-linear constraint equations (16 and 17) are obtained⁵. An iterative procedure is employed to solve the system of non-linear equations. At each iteration, the denominator p_{cs} is held constant using the previous estimate of \vec{p} and the resulting linear system of equations is solved. The iterative procedure is repeated until there is convergence.

⁴Due to noise both in image measurements and pose estimates.

⁵A minimum of two frames is needed to solve the system of equations.

In practice, we have found one iteration is sufficient for robust results. The input covariance matrix V required for the normal equations is obtained from the expressions derived above for the noise terms ζ_x, ζ_y . The output covariance of the 3D point estimate can also be computed.

In the batch method, information from all frames is used simultaneously to estimate the 3D locations of tracked image points. However, it may be desired to sequentially update the location of new points after every pair (or a larger set) of frames. In the sequential or quasi-batch mode, equations (6 and 7) are again used to estimate the 3D location of image points tracked over the current set of frames. These new estimates must be fused with the previous estimates to obtain the current optimal estimate. The covariance matrices associated with each estimate are used to fuse the two estimates and provide a new uncertainty matrix using the standard Kalman Filtering equations.

Let the estimate of the point's 3D location and its covariance at frame " t_1 " be $\tilde{p}(t_1)$ and $\Lambda_p(t_1)$ respectively. A new 3D location measurement \tilde{Q} with uncertainty (covariance matrix Λ_Q) is computed from a batch of " n " image frames. The fused location estimate $\tilde{p}(t_n)$ and updated covariance matrix $\Lambda_p(t_n)$ at frame " t_n " are given by:

$$\tilde{p}(t_n) = \Lambda_p(t_n)(\Lambda_p(t_1)^{-1}\tilde{p}(t_1) + \Lambda_Q^{-1}\tilde{Q}) \quad (20)$$

$$\Lambda_p(t_n) = (\Lambda_p(t_1)^{-1} + \Lambda_Q^{-1})^{-1} \quad (21)$$

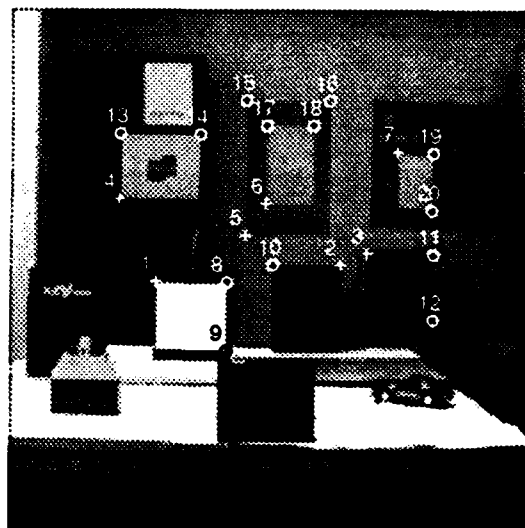
This same method is used for model refinement. Initial model points have associated with them their input covariance matrices. When the model is tracked over a new batch of frames, 3D measurements can also be made for the model points by the above pseudo-intersection procedure. These new measurements are fused with the old estimate using the above equation.

5 Experimental Results and Discussion

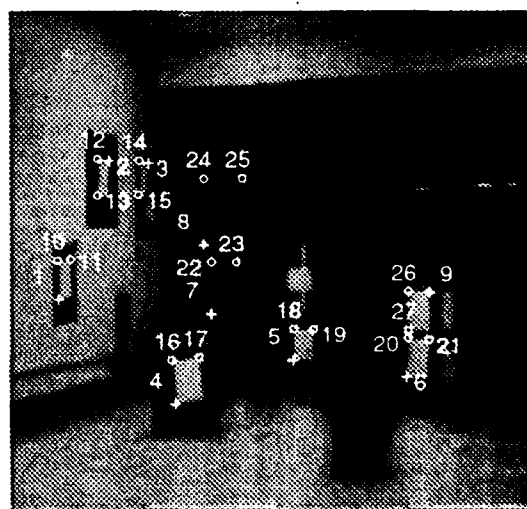
We now present results on two multi-frame sequences. In both cases, similar results are presented using an initial model built from points on the recovered shallow structures. The image sequences were captured with a SONY B/W AVC-D1 camera, with an approximate FOV of 24 degrees and digitized to 256-by-242 pixels. In all experiments the image center was assumed to be at the center of the image frame and the effective focal length was calculated from the manufacturers specification sheets. Since we have shown in [Kumar, 1990] that errors in the image center do not significantly affect the location of new points in a world coordinate system (for a small field of view imaging system), calibration for the image center has not been done.

The A211 (10 frames) and COMP (6 frames) sequences were generated by taking images from a camera mounted

on a mobile robot moving roughly parallel to the optical axis. Figure 2 shows the first frames in the A211 and COMP sequences. Between consecutive frames, the robot was translated approximately 0.38 and 1.4 feet respectively for the A211 and COMP image sequences. The depth of some salient structures in each sequence was measured with a tape measure.



(a)



(b)

Figure 2(a) A211 and 2(b) COMP Images.

In both sequences, image lines were extracted for each frame using Boldt's [Boldt, et al, 1989] line grouping system. The tracking algorithm was applied to the image sequences to identify the shallow structures in the scene. Line triples were automatically selected to hypothesize aggregate structures. Each of these was tested for affine trackability, resulting in its labeling as a shallow or a non-shallow structure [Sawhney, 1992a]. Figure 3 shows in bold lines the structures identified as shallow by the algorithm for the two sequences.

The numbered points marked by crosses in Figure 2 lying on the recovered shallow structures were used as the initial model points for the A211 and COMP image sequences respectively. These points are defined by the intersection of some of the pairs of lines belonging to shallow structures. The 3D model locations were constructed by back projecting the points in the first image's coordinate frame.

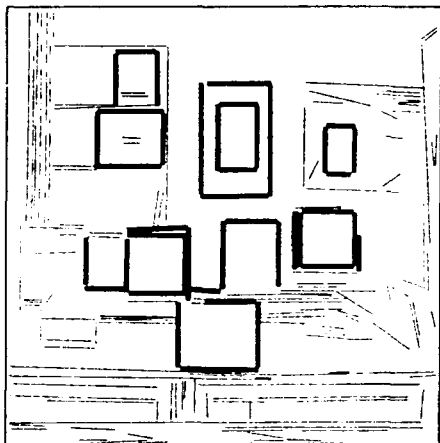


Figure 3. Shallow structures indentified in the A211 and COMP image sequences.

The model extension and refinement algorithm was run in a sequential mode. Tables 1 and 2 show the result of locating new points (circled and numbered in Figure 2) and refining the initial model points (marked by crosses in the figures). The ground truth available for both experiments was only the depths (as opposed to 3D location) of the points in the first image's coordinate frame. Thus the results shown in Tables 1 and 2 com-

pare the measured depth value (ground truth) with the recovered depth value. Column 2 in the tables shows the measured depth of the point in the first image coordinate frame. Columns 3 and 4 show the error and percentage error in depth, respectively, for the initial model points as acquired by the affine-based tracking algorithm. Columns 5 and 6 show the output error and percentage error in depth (after model refinement and extension) respectively.

For the new points, it is assumed that no initial model was available; therefore columns 3 and 4 for these points are blank. Note that these points also belong to the reconstructed shallow structures. However, their reconstructed locations were not used as a part of the initial partial model. Instead, these points were used to demonstrate model extension because the ground truth was available only for these structures. In the tables, the percentage error in depth is computed with respect to the depth in the first image's coordinate frame.

Table 1: Absolute and Percentage 3D location errors for points in A211 sequence (see Fig. 2.)

		INPUT		OUTPUT	
Pt. No.	Depth ft.	Abs. Err. ft.	% Err.	Abs. Err. ft.	% Err.
Initial Points					
1	13.4	0.24	1.80 %	0.24	1.78 %
2	14.6	0.19	1.31 %	0.20	1.34 %
3	19.0	0.74	3.88 %	0.66	3.46 %
4	19.0	0.16	0.86 %	0.11	0.60 %
5	20.4	0.13	0.62 %	0.17	0.86 %
6	20.4	0.39	1.90 %	0.32	1.60 %
7	20.4	0.49	2.38 %	0.46	2.25 %
New Points					
8	13.4	-	-	0.11	0.79 %
9	13.4	-	-	0.00	0.01 %
10	14.6	-	-	0.53	3.65 %
11	19.0	-	-	0.73	3.86 %
12	19.0	-	-	0.54	2.82 %
13	19.0	-	-	0.11	0.59 %
14	19.0	-	-	0.07	0.34 %
15	20.4	-	-	0.23	1.13 %
16	20.4	-	-	0.27	1.32 %
17	20.4	-	-	0.12	0.57 %
18	20.4	-	-	0.34	1.65 %
19	20.4	-	-	0.62	3.02 %
20	20.4	-	-	0.59	2.92 %
Average depth error of new points					1.63 %

The average input error in depths of the seven initial model points in the A211 sequence (as recovered by the affine-based tracking algorithm) was 0.4 feet (1.85 % error). At the end of the ten frames, the average error of

Table 2: Absolute and Percentage 3D location errors for points in COMP sequence (see Fig. 2.)

		INPUT		OUTPUT	
Pt. No.	Depth ft.	Abs. Err. ft.	% Err.	Abs. Err. ft.	% Err.
Initial Points					
1	29.3	-0.23	0.80 %	-0.11	0.36 %
2	31.3	0.26	0.84 %	0.17	0.55 %
3	34.2	-0.10	0.29 %	-0.07	0.20 %
4	25.7	-0.26	1.03 %	-0.23	0.88 %
5	35.8	1.59	4.43 %	1.54	4.31 %
6	28.7	0.39	1.36 %	0.39	1.35 %
7	43.2	-1.65	3.82 %	-1.63	3.76 %
8	43.2	1.18	2.73 %	1.15	2.66 %
9	28.7	1.46	5.08 %	1.41	4.91 %
New Points					
10	29.3	-	-	0.25	0.86 %
11	29.3	-	-	-0.35	1.19 %
12	31.3	-	-	0.51	1.63 %
13	31.3	-	-	0.28	0.89 %
14	34.2	-	-	0.93	2.70 %
15	34.2	-	-	1.31	3.82 %
16	25.7	-	-	-0.02	0.07 %
17	25.7	-	-	0.03	0.11 %
18	35.8	-	-	1.05	2.93 %
19	35.8	-	-	0.50	1.40 %
20	28.7	-	-	-0.11	0.39 %
21	28.7	-	-	0.08	0.29 %
22	43.2	-	-	0.46	1.07 %
23	43.2	-	-	1.77	4.10 %
24	43.2	-	-	-0.45	1.04 %
25	43.2	-	-	0.13	0.30 %
26	28.7	-	-	0.80	2.77 %
27	28.7	-	-	0.25	0.88 %
Average depth error of new points					1.46 %

the 7 initial points was 0.37 feet (1.76 %). The thirteen new points were located to an average accuracy of 0.4 feet (1.63 %).

The average input error in depths of the nine initial model points in the COMP sequence (as recovered by the affine-based tracking algorithm) was 1.01 feet (2.27 % error). At the end of the six frames, the average error of the 9 initial points was 0.98 feet (2.11 %). The eighteen new points were located to an average accuracy of 0.69 feet (1.46 %). For this experiment the measured depth values are only approximate to about 0.5 feet for some points. This is especially true for points lying on the left side wall (points 1, 2, 3 etc. in Figure 5).

In both experiments, the model extension process was fairly accurate in locating new points. However, there was only slight improvement in the initial model as a

result of the model refinement process. The robust recovery of the location of new 3D points depends on the camera motion. Optimal angles for triangulation are achieved when there is significant translation parallel to the image plane. In the A211 and COMP sequence, the translation of the camera is mostly along the optical axis. Thus, the FOE (focus of expansion) lies on the image plane. Points close to the FOE have smaller disparity and their depths cannot be reliably estimated. Consequently, these results imply that we may be at the limit of recoverable accuracy.

Finally, the accuracy of the model extension process depends on the initial accuracy of the model points. If the initial model points have a large amount of noise, then the poses determined for any batch of frames will be highly correlated. In this case, the 3D location estimates of new points will be correlated both across all points and also all frames. To fully account for this OBcorrelation, covariance matrices equal to the size of number of points times number of frames will have to be inverted. In our case, it is assumed that the initial points do not have significant noise and hence the cross-correlations can be ignored. But for larger amounts of noise, it may not be possible to ignore these effects [Oliensis, 1991].

Appendix

Some facts from linear system estimation theory are reviewed. An unknown parameter vector \vec{x} with "p" elements is related to a set of "n" noisy observations \vec{y} by the following equation:

$$A\vec{x} = \vec{y} + \vec{\eta} \quad (22)$$

where $\vec{\eta}$ is zero-mean Gaussian noise with covariance matrix V . Assume, that this set of equations is an over-constrained system. Then the Best Linear Unbiased Estimate (BLUE) [Strang 1986] of the unknown vector \vec{x} and the covariance matrix "P" of the output parameters are given by:

$$\hat{\vec{x}} = (A^T V^{-1} A)^{-1} A^T V^{-1} \vec{y} \quad (23)$$

$$P = (A^T V^{-1} A)^{-1} \quad (24)$$

References

- [1] M. Boldt and R. Weiss and E. Riseman, "Token-based Extraction of Straight Lines," *IEEE Transactions on Systems Man and Cybernetics*, volume 19, no. 6, pp. 1581-1594, 1989.
- [2] T. J. Brodia and R. Chellappa, "Estimating the Kinematics and Structure of a Rigid Object from a Sequence of Monocular Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 497-513, 1991.
- [3] S. Chandrasekhar and R. Chellappa, "A Two-Step Approach to Passive Navigation Using a Monocular Image Sequence," *USC-SIPI Technical Report*

170, University of Southern California, Electrical Engineering-Systems, 1991.

- [4] N. Cui, J. Weng and P. Cohen, "Extended structure and motion analysis from monocular image sequences," *Proceedings 3rd IEEE International Conference on Computer Vision*, Osaka, Japan, Dec. 1990.
- [5] R. Kumar and A.R. Hanson, "Robust Estimation of Camera Location and Orientation from Noisy Data with Outliers," *Proc. IEEE Workshop on Interpretation of 3D scenes*, Austin, Texas, Nov. 1989.
- [6] R. Kumar and A.R. Hanson, "Sensitivity of pose refinement to accurate estimation of camera parameters," *IEEE International Conference on Computer Vision*, Osaka, Japan, Dec. 1990.
- [7] P. C. Mahalanobis, "On the Generalized Distance in Statistics," *Proceedings of the National Institute of Science, India*, (12), pp. 49-55, 1936.
- [8] J. Oliensis and J. I. Thomas, "Incorporating motion error in multi-frame structure from motion", *Proceedings IEEE Workshop on Visual Motion*, Princeton, N.J., Oct. 1991.
- [9] H.S. Sawhney, "Spatial and Temporal Grouping in the Interpretation of Image Motion", Ph.D. Thesis, Computer Science Department, University of Massachusetts (Amherst), 1992(a). Available as Technical Report TR92-05.
- [10] H. S. Sawhney and A. R. Hanson, "Comparative Results of Some Motion Algorithms on Real Image Sequences," *Proc. DARPA Image Understanding Workshop*, 1990.
- [11] H. S. Sawhney and A. R. Hanson, "Identification and 3D description of 'shallow' environmental structure in a sequence of images", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 179-186, Hawaii, June 1991.
- [12] H. S. Sawhney and A. R. Hanson, "Affine Trackability aids Obstacle Detection", *Proc. CVPR*, Champaign, IL, June 1992(b), pp. 418-424.
- [13] Gilbert Strang, "Introduction to Applied Mathematics," Wellealey-Cambridge Press, MA, 1986.
- [14] L. R. Williams and A. R. Hanson, "Translating Optical Flow into Token Matches and Depth from Looming", *Second Int. Conf. on Computer Vision*, pp. 441-448, 1989.

Range-Free Qualitative Navigation*

David Dai and Daryl T. Lawton
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

Abstract

We summarize some recent algorithms developed for qualitative navigation which are completely independent of range estimates to landmarks. We introduce several distinctions that reflect more realistic application of qualitative navigation algorithms to real robots. These involve the extent to which landmarks can be identified from very different points of view (called the *distinctiveness* of landmarks); whether or not a compass is allowed; and distinctions between different types of compasses.

1 Introduction

Qualitative Navigation [Kuipers and Byun, 1987; Kuipers, 1978; Levitt and Lawton, 1990] concerns spatial learning and path planning in the absence of a single global coordinate system for describing locations and the positions of landmarks. It is based on a multi-level representation of space, which, at its most abstract level, is based on topological properties which allow a robot to describe a location using the directions of visually salient patterns (with no associated range measurements) and then navigating using cues such as the occlusions that occur between landmarks. An advantage is that the robot can use landmarks for which exact positions can not be determined. Thus, if a robot sees a building in the distance, it may not know or be able to recognize the structure as a building or determine its exact position in space but it can still incorporate this to form an effective spatial memory. This is actually quite intuitive: it is doubtful that animals navigate by detecting landmarks, determining ranges to them, and then storing everything in a single frame of reference [Gallistel, 1990]. It also removes the effects of incremental errors due to drift.

Our work [Levitt and Lawton, 1990] in qualitative navigation developed while trying to produce basic navigation and recognition capabilities in an autonomous land vehicle. Initially we worked with a terrain representation based upon an *a priori* terrain grid, which describes terrain in terms of a regular grid of features

referenced with respect to a single global coordinate system. We discovered several problems with such spatial representations. The grids would describe large patches of terrain by a set of numbers which corresponded to terrain features such as elevation and vegetation type. Unfortunately, the world consists of objects which are difficult to summarize by a single set of numbers. It is difficult to establish the exact three-dimensional position of a distant landmark, especially when using passive sensing. Thus, it is difficult to know where to attach landmarks to the underlying terrain representation when it uses a single, global coordinate system. Robots also have limited recognition capabilities in complex outdoor environments. They can see distinctive things in the world, and yet not know what or where they were. In fact, there are no assurances that robots can see the same object as being the same object from very different points of view.

Qualitative Navigation deals with these problems via a multi-level representation of spatial memory. The different levels are distinguished by what constitutes a landmark and the connectedness of spatial memory which refers to how, given one location, it is possible to determine the position of another location. At the simplest level of spatial representation (the *Sensorimotor level*) a landmark consists of a perceptual event which can be used for sensory feedback to control guidance. The next level (the *Topological level*) is based upon noting and tracking stable perceptual events around the robot, but without associating any range information to these. This level is topological in the sense that there is no metric information associated with landmarks. A place is described by the set of visual patterns surrounding the robot. This description of a place is called a *viewframe*. Movement from place to place is determined when there is some change in the order of these patterns. The next level allows the association of potentially inexact range information with the visual patterns (*Local Coordinate systems*). At this level, viewframes can have associated range estimates with the detected visual patterns and the localization of one place to another was inexact. The final level (*Global Coordinate System*) assumes that we have exact three-dimensional information for all landmarks. In [Levitt and Lawton, 1990], we found that by working at the level of a viewframe based representation, the problems

*This research is supported by the Advanced Research Projects Agency of the Department of Defense and is monitored by the U. S. Army Topographic Engineering Center under contract No. DACA76-92-C-0016

faced when working with a single global coordinate system were drastically simplified.

In this paper we describe qualitative navigation algorithms which work completely at the topological level, dealing with landmarks for which there are no range estimates. In addition, we introduce several distinctions for qualitative navigation algorithms. One distinction concerns landmarks. We consider two basic types: **distinct landmarks** which can always be recognized as the same from wherever they are seen and **nondistinct landmarks** which may not be recognized as being the same when seen from different points of view. We assume that once landmarks are seen, they can be tracked over time until they disappear. The other distinction involves whether or not the navigation algorithms use a compass to yield a fixed direction. We also distinguish two different types of compass. The direction associated with a **local compass** can change from place to place, but at a given place, it will always point in the same direction. An example is a compass which is effected by fixed magnetic influences at different locations. The local compass can also be a very strong landmark which is visible from a wide set of views. A **global compass** will always point in the same direction regardless of where the robot is located. We can express these distinctions as a table corresponding to the different types of topological navigation algorithms we have developed:

Topological Qualitative Navigation Algorithms

	Compass	No Compass
distinct landmarks	Very Good	Good
nondistinct landmarks	Good	Difficult!

For example, consider qualitative navigation without a compass and identical, nondistinct landmarks. As one might expect, this is very difficult and depends critically on matching viewframes based exclusively upon the angular orientations of landmarks. More practical algorithms are those which are based upon the use of a local compass and a limited number of distinct landmarks. This corresponds to a freely navigating robot which can build maps and navigate using simple visual features, such as colored regions aligned with gravity, as landmarks.

In the remainder of this paper, we describe the basic memory organization used for qualitative navigation and then present different navigation algorithms.

2 Organization of Spatial Memory and Navigation Behaviors

2.1 Landmarks

We distinguish between types of landmarks to reflect different recognition capabilities in robots. A **distinct landmark** is one which can be recognized as being the same from all points of view. Distinct landmarks require considerable recognition capabilities for a robot owing to the variable appearance of landmarks from different points of view. A **nondistinct landmark** is one which may not be recognized as being the same from different points of

view. We assume that once a nondistinct landmark is seen, it can be tracked over time until it disappears. A nondistinct landmark is not necessarily described as a particular object in the world, but can be described as a simple visual pattern, such as a colored region of a particular shape or a set of edges aligned with gravity. Such descriptions of landmarks will tend not to be unique.

A general finding of the algorithms we describe here is that the more distinct landmarks there are, the more easily a robot can find shortcuts and novel paths between locations. The more indistinct landmarks there are, determining position depends on recognizing the distribution of landmarks surrounding a robot. In this case, the robot will tend to stay close to established paths that it determines during explorations. It is possible for a robot to determine novel paths between locations with nondistinct landmarks, but it requires significant exploration to determine that a landmark is the same form many different points of view.

2.2 Viewframes

A **viewframe** contains the set of visible landmarks surrounding a robot at a given location with their corresponding orientations and other attributes describing the individual landmarks (such as color, visible height, contrast, etc.) Viewframes are a one-dimensional sequence of landmarks (The direction of gravity is used to reduce the two-dimensional images surrounding the robot to a one-dimensional sequence). An example viewframe V is shown in Figure 1. This viewframe uses compass information and is then represented as

```
[Viewframe Identifier:  V
Landmarks:
[[lidA; AttributesA], αA]
[[lidB; AttributesB], αB]
[[lidC; AttributesC], αC]
Robot's heading:      αh]
```

When a viewframe is extracted without a compass, there's no associated 0-axis to describe a fixed direction. The relative orientation of landmarks is then represented by the angle difference between successive landmarks. The same viewframe in Figure 1 is represented (shown in Figure 2) as

```
[Viewframe Identifier:  V
Landmarks:
[[lidA; AttributesA], θA]
[[lidB; AttributesB], θB]
[[lidC; AttributesC], θC]
Robot's heading:      (θh, B)]
```

For the viewframe in Figure 1 and Figure 2, lid_A , lid_B , lid_C are the **local identifiers** for visible landmarks A , B , C . The **Local identifier** is a name or abstraction of the attributes of a landmark that is tied to a specific viewframe. Note that a landmark with the same local identifier in different viewframes can have different image attributes depending upon the viewframe it is contained in. A distinct landmark which can be recognized as being the same from very different points of view has a

unique local identifier with respect to all viewframes and is called a **global identifier**. When a robot is exploring the environment, distinct landmarks will always be associated with a unique local identifier in all the viewframes which contain it. Nondistinct landmarks will have the same local-identifiers in connected viewframes so long as the landmark is visible (or after landmark-unification – see below). When a landmark reappears or is disoccluded, it will have a new associated local identifier. This is similar to what can happen when an animal walks on two different paths without realizing that there is a common landmark between them. For each nondistinct landmark, there can be more than one local identifier for it in different viewframes.

When viewframes consist largely or totally of nondistinct landmarks, being able to access or recognize a particular viewframe is difficult (for example, a large red region can be a landmark in several different viewframes). For this reason, we also associate keys with viewframes that are used for recognizing viewframes by a hashing operation. There are a large number of different keys such as the average height of landmarks, the average angle between landmarks, the number of landmarks, number of highest landmarks, number of landmarks for particular colors, variance of contrasts, variance of heights, variance of angles between landmarks, ratio's of landmarks having different attributes, etc. These keys help to distinguish and match viewframes. If there is a local compass many more types of keys are possible because it is possible to order the landmarks in the viewframe and compute keys based upon position in the viewframe. Each key has limited number of values. Two viewframes are said to be *hash-matched* if they have the same key value for each key.

Keys are also useful for the efficiency of accessing viewframes. Suppose we have 10 keys, each key has 10 different values; therefore we have 10^{10} equivalence classes. We build a hash table, making an entry for each possible value combination of all keys. To find a viewframe to match V , we first compute key values of V for all keys, then use the combination of those values as an index to the hash table to find the viewframe in the database. Since we have $O(1)$ number of keys, time to compute the key value is $O(1)$, time to search in the hash table is $O(1)$; therefore, the time complexity to find a viewframe to match V is reduced to $O(1)$.

2.3 Viewframe Extraction and Filtering

The extraction of a viewframe involves identifying landmarks surrounding the robot. These are then stored in different types of viewframes depending upon whether or not there is a compass and on the distinctiveness of the landmarks. We have also found it useful to compare a newly extracted viewframe to the previously extracted viewframe to determine if the newly extracted viewframe is different or novel enough to merit storing it in spatial memory. This process is called *viewframe-filtering* and has the effect of reducing the number of very similar or redundant viewframes that are stored in memory. Filtering is done by keeping track of changes in the values associated with the different keys. There is a threshold

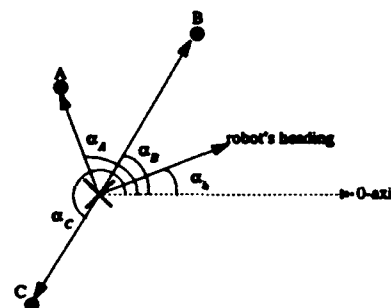


Figure 1: Viewframe Representation with a Compass

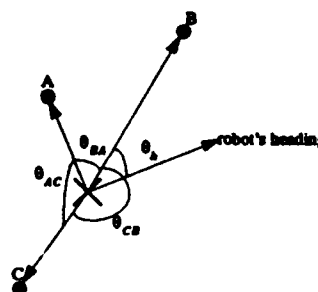


Figure 2: Viewframe Without a Compass

associated with allowable changes in the value for each key. If this is exceeded, then the viewframe is stored in spatial memory. For example, if there number of landmarks changes drastically, it is necessary to then extract a viewframe in spatial memory. It may also be useful to have a function which weights the changes in the different keys to determine whether a viewframe is novel enough to be extracted.

2.4 ViewFrame Matching

Viewframe matching is the process which determines the similarity of two viewframes. We use a two level matching processing. The first level finds similar viewframes by hashing and then uses the number of landmarks with common local identifiers in both viewframes as a measure of similarity called *connectivity*. First level connectivity between two viewframes is defined as:

$$con(V_1, V_2) = \frac{\|Local_ids(V_1) \cap Local_ids(V_2)\|}{\|Local_ids(V_1) \cup Local_ids(V_2)\|} \quad (1)$$

Second level viewframe matching compares the orientation (angle) difference between landmarks. For this level of matching, different thresholds for the maximum orientation difference for corresponding local identifiers in the viewframes are used.

2.5 Navigation Behaviors

The navigation algorithms are based on a set of simple visual tracking behaviors. **Viewframe centering** is when the robot is positioned at a landmark and walks in the direction of the center of a viewframe which contains that landmark. Without a compass, viewframe centers

involves moving so that the robot optimizes key similarity with respect to the viewframe. Viewframe centering is simpler with a local compass which is valid withing the extracted viewframe since the relative direction of a landmark and the viewframe center is known. **Viewframe back-matching** (also called **landmark unification**) involves recognizing that landmarks in different viewframes are the same and their local identifiers are unified. This happens when a robot visits the same place along separate paths. **Landmark circling** is when a robot circles around a known landmark. It is a way of searching for surrounding landmarks when no nearby landmarks are distinguished or visible. The robot can spiral towards or away from the landmark (until the landmark is no longer visible). **Landmark targeting** is for walking towards a visible landmark. **LBP crossing** is when a robot crosses an Linear Pair Boundary defined by two landmarks. The crossing can occur on either side or through the center of the LPB between the two landmarks. **LBP alignment** is when the robot travels along the LPB boundary defined by two landmarks. **Random walking** randomly selects a visible landmark; walks to it; and then repeats. An alternative version walks straight for some distance, changes direction, and then repeats. In **Novelty walking**, a robot walks to optimize the changes in the keys used for viewframe filtering. The effect is to go someplace where it is as different as possible from where you currently are.

2.6 Spatial Memory

Spatial memory consists of three inter-related databases: the viewframe database (V-DB), the path database (P-DB) and the landmark database (L-DB) (see Figure 3). The landmark database contains descriptions of landmarks that a robot has seen. It is possible for the same physical landmark to occur several different times in the landmark database because it may not have been identified as being the same from different views. The viewframe database contains viewframes which describe the visible landmarks surrounding a robot at a given location (this is described in more detail shortly). The path database consists of connected sequences of viewframes which a robot determines while exploring the environment. Database Storage algorithm:

Step 1 extract *VF* and filter against previously extracted *VF*.

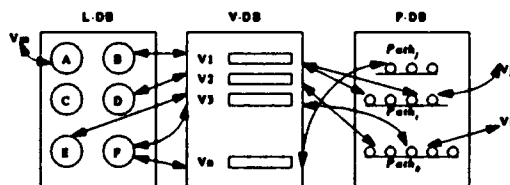


Figure 3: Memory Architecture

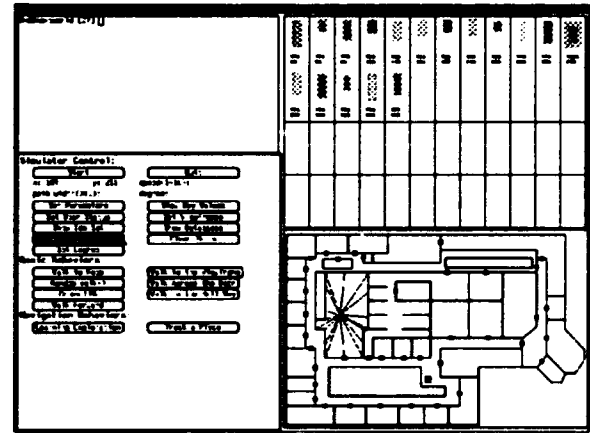


Figure 5: Simulator for Indoor Robot with displayed viewframe

Step 2 compare *VF* with other viewframe in V-DB by some viewframe matching mechanism.

Step 3 if *NOT* matched for *VF*, add *VF* to V-DB, add pointer to *VF* into each landmark entry with the same local identifier in L-DB; else return the pointer to *VF* in V-DB.

Step 4 add pointers to *VF* into current path's viewframe sequence in P-DB;

2.7 Qualitative Navigation Simulator

We have been exploring different qualitative navigation schemes using the simulators shown in (Figure 4) and (Figure 5) (for exploring indoor navigation). Each contains 4 subwindows. The upper-left is a Unix shell; the lower-left has controls for setting the such things as the density of landmarks, the range of visibility, the number of globally distinct landmarks, selecting different navigation modes and so forth; and the upper-right shows the 360 degrees of view from the robot at a given location. The lower-right shows a top-down view of the navigation world. In (Figure 4) the circle shows current viewframe containing landmarks displayed in upper-right subwindow, the line in the circle shows the robot's heading. Distinct landmarks are numbered and nondistinct landmarks are not numbered, but can appears as having different colors and intensities. In the simulator in (Figure 5), we assume that limitations on sight are only caused by occlusion. The current viewframe is shown as the set of radiating lines from the robot's current position to each of the visible landmarks. The current viewframe is displayed as a sequence of landmarks in the upper right-hand window.

3 Navigation Using A Local Compass With A Variable Percentage Of Distinct Landmarks

This algorithm is intended to work with a variable number of distinct landmarks, ranging from completely nondistinct landmarks to completely distinct landmarks. The nondistinct case would correspond to walking

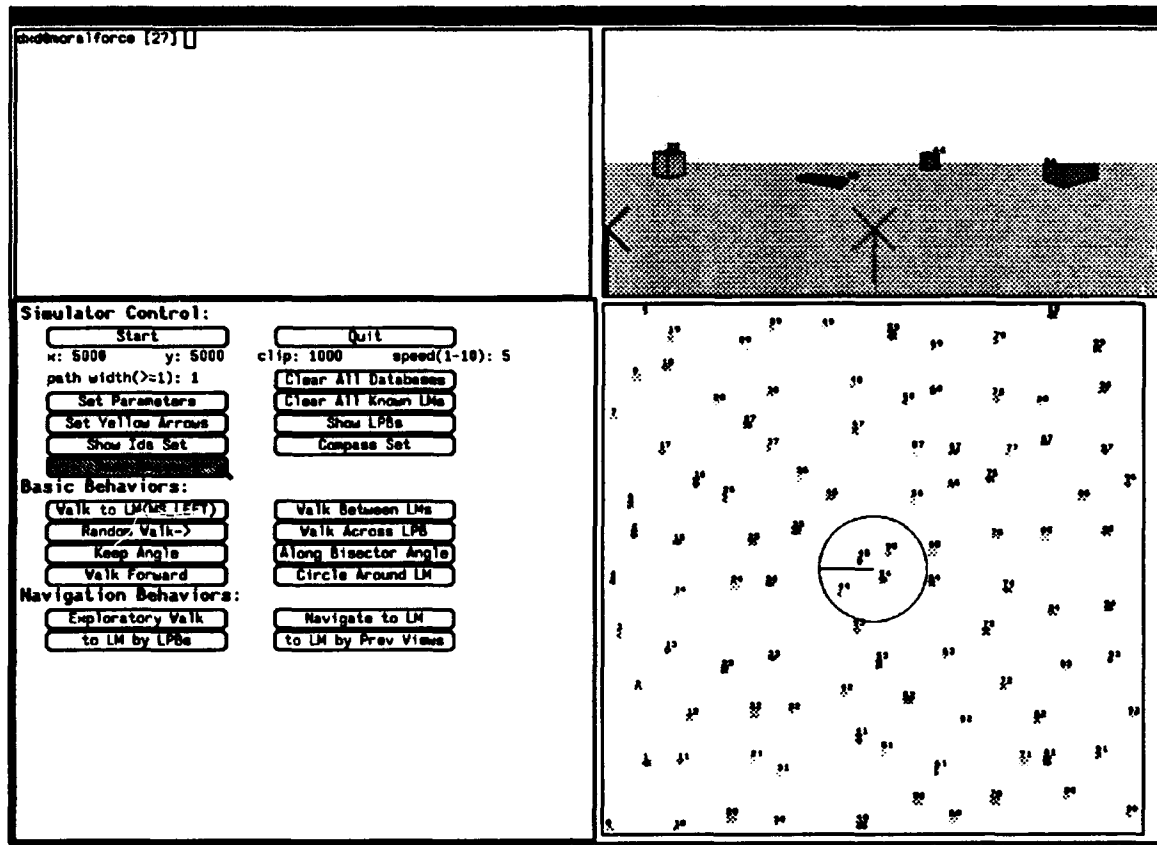


Figure 4: Qualitative Navigation Simulator

through a world full of identical landmarks with a compass. When the number of distinct landmarks increases, the efficiency of the path planning improves.

Navigation using this algorithm is shown in figures 7, 8, and 9. The robot initially walks along two separate paths which form a Ψ -like shape shown by the solid thin lines. The robot is first at the upper-middle part of the Ψ and walks to the middle-lower part. It is then relocated to the upper-left corner and walks to the upper-right corner along a curved path. As it walks along these paths, it keeps track of landmarks and stores extracted viewframes in the different system databases. It then has the task of going from the upper-right corner of Ψ to the upper-left corner of Ψ . To do this, the robot can either follow the long curved path that it originally followed or else it can find a short-cut directly between them. The key result is that as the number of distinct landmarks increases, the robot is able to find increasingly more direct paths between locations. With more nondistinct landmarks, navigation involves staying close to paths that have been previously followed. Shortcuts are possible when common landmarks between paths are found.

The algorithm utilizes viewframe centering and viewframe back-matching. In **Viewframe centering** a robot walks from a landmark towards the center of a viewframe which contains that landmark. If the robot is at landmark with local identifier L and orientation angle

α in V , viewframe centering is to walk in orientation angle $\alpha + \pi$ towards the center of V . Two viewframes are said to be *connected* or *adjacent* if they have at least one local identifiers in common. Navigation then involves finding a sequence of connected viewframes ($V_0, \dots, V_j, \dots, V_n$) with overlapping landmarks which are traversed by successive viewframe centering.

Viewframe back-matching (also referred to as landmark unification) is used to determine that landmarks having different local identifiers in different viewframes actually are the same physical landmark. They can then be used to navigate from one viewframe to another and form the basis of finding shortcuts when such common landmarks are recognized. During viewframe centering to V_j , if the robot cannot find a *distinct* landmark in common between V_j and V_m ($m > j$ and $m < n$), it attempts viewframe back-matching to update local identifiers in the viewframe database. This is illustrated by Figure 6. The robot is currently at V_B and has previously extracted V_A with local identifiers L_1, L_2, L_3 associated with the nondistinct landmarks. V_A and V_B have local identifier L_1 in common. The robot first goes to landmark L_1 , then walks towards the center of V_A . While it is walking towards the center of V_A , it continues to extract viewframes and perform second level viewframe matching (based upon angle and orientations of landmarks) with respect to V_A . When it extracts a viewframe at C (which is nearby A) with new local iden-

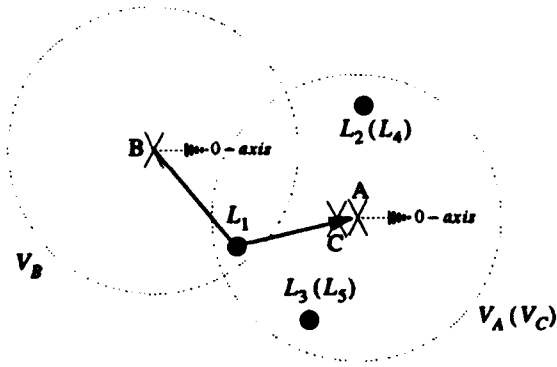


Figure 6: Viewframe Back-Matching with a Compass

tifiers L_4 for L_2 , L_5 for L_3 , the robot matches V_C to V_A , updating the viewframe database by substituting L_3 into L_5 , L_2 into L_4 .

The algorithm has the following steps:

Goal: A landmark with local identifier $lcid$ (tied to a specific viewframe) to go to.

Step 0 If $lcid$ is in current viewframe, go directly to it and the algorithm terminates.

Step 1 Create a *virtual* viewframe V_n containing only the goal $lcid$ with an unspecified orientation. Construct an N by N weight matrix W (N is current total number of viewframes plus one). For each pair of viewframes (V_i, V_j) including the *virtual* viewframe V_n and current starting viewframe V_0 , compute $con(V_i, V_j)$ by equation (1), if it is greater than a certain threshold (we select 0), we assign the weight matrix entry $W(i, j) = 1$; otherwise $W(i, j) = \infty$. With the weight matrix, we find a *least* sequence of connected viewframes V_0, V_1, \dots, V_n by applying a shortest path algorithm[Cormen *et al.*, 1990]. Alternatively, if the total number of viewframes N is too great, we use a breadth-first tree search[Cormen *et al.*, 1990] from V_0 to find adjacent viewframes, such that a viewframe cannot appear twice in a path of the tree.

Step 2 If a sequence of *connected* viewframes are not found, stop. Otherwise the robot performs *viewframe centering* and *viewframe back-matching* through V_0, V_1, \dots, V_n . It walks to the landmark with common local identifier in both V_0 and V_1 , where choice of distinct landmark has priority.

Step 2.1 If the robot is currently at landmark P of viewframe V_i (i is max), it *viewframe centers* towards the center of V_i , testing if current viewframe V_c is adjacent to V_m , i.e. $m \in [i+1, n]$ and m is max such that $con(V_c, V_m) > 0$; if m is found which means a distinct landmark is found, then it changes the direction and walks to the landmark in both V_c and V_m . Otherwise, it performs *back-matching* to V_i ; if no ambiguity occurs and the best match is found, the robot updates the local identifiers, i.e. it uses local identifiers in V_i to replace

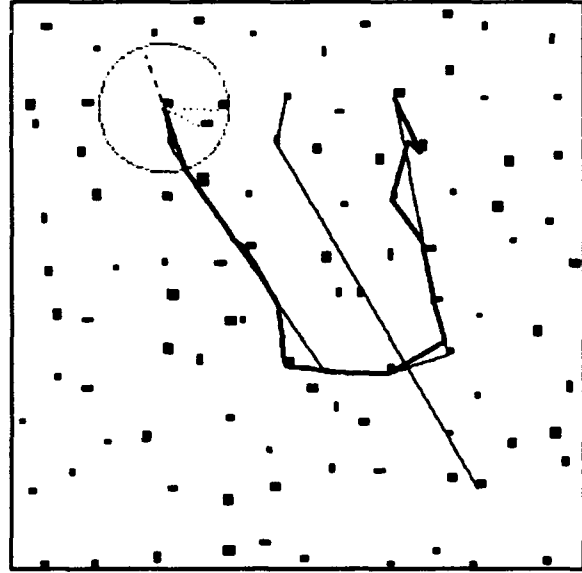


Figure 7: 100% Nondistinct Landmarks. Path Planning (Solid Thick Path) from Upper-Right Corner to Upper-Left Corner of the Ψ

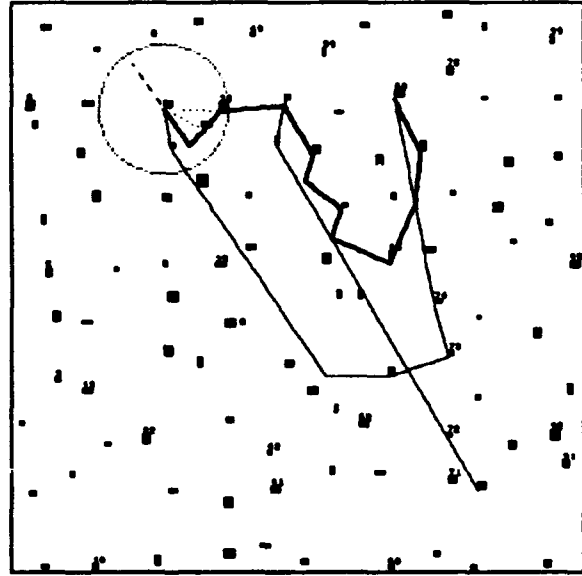


Figure 8: 75% Nondistinct Landmarks. Path Planning (Solid Thick Path) from Upper-Right Corner to Upper-Left Corner of the Ψ

corresponding local identifiers with the same orientations in V_c as well as those local identifiers in V-DB; and then walks to a landmark with common local identifier both in V_i and V_{i+1} .

Step 2.2 Repeat Step 2.1 until the goal is achieved, or failure due to ambiguity.

When all the landmarks are distinct, viewframe back-matching is unnecessary.

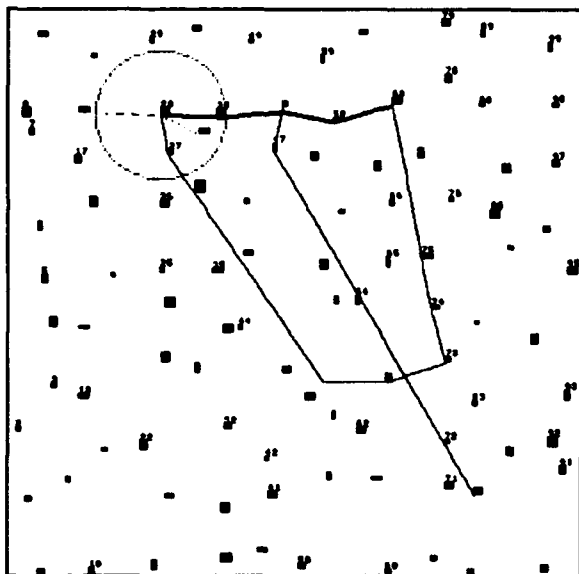


Figure 9: 50% to 0% Nondistinct Landmarks. Path Planning (Solid Thick Path) from Upper-Right Corner to Upper-Left Corner of the Ψ

4 Navigation Without a Compass for Distinct Landmarks (No LPBs)

This algorithm assumes distinct landmarks and no compass and no landmark pair boundaries (LPBs). LPB-based navigation is described in next section.

The algorithm relies on **viewframe circling** to compensate for the lack of a compass. Figure 10 shows an example of navigation using this algorithm with the viewframes and paths from the previous figures. The exploration paths Ψ (solid thin lines) are generated in the same manner as in figures 7, 8, 9. The path determined by the robot is shown as a solid thick line from the upper-right corner of the Ψ to the upper-left corner of the Ψ . The result shows that, the path is slightly longer than that with a compass (in Figure 9). The processing example in figures 11, 12, 13, 14, shows some of the interesting characteristics of this algorithm. In Figure 11, the robot moves to Landmark 89 by an exploratory behavior to generate a path. We then want the robot to walk back-and-forth between Landmark 89 and 64 to find increasingly more direct paths. Initially, the robot can not find, due to limitations on its range of vision, most of the visible landmarks along its original path. So it begins to circle around the current landmark to search for landmarks from the path it traversed. This continues until it returns to its origin. The circling behavior for finding landmarks is responsible for the indirect looking paths. As the robot traverses back-and-forth between landmarks 64 and 89, it is able to use the viewframes it stored from previous trips to determine a more direct path. The robot will determine different paths between the two landmarks depending upon the direction in which it travels. This is because the robot can not see the same landmarks when traveling in the different directions due to limitations on allowable view-

ing distance. The further the robot can see, the more direct and similar the paths found under this algorithm become.

The algorithm has the following steps:

Goal: A landmark with identifier id to go to.

Step 0 If id is in current viewframe, go directly to it and the algorithm terminates.

Step 1 Create a *virtual* viewframe V_n containing only the goal id with arbitrary orientation. Construct an N by N weight matrix W (N is current total number of viewframes plus one). For each pair of viewframes (V_i, V_j) including the *virtual* viewframe V_n and current starting viewframe V_0 , compute $con(V_i, V_j)$ by equation (1), if it is greater than a certain threshold (we select 0), we assign the weight matrix entry $W(i, j) = 1$; otherwise $W(i, j) = \infty$. With the weight matrix, we find a *least* sequence of connected viewframes V_0, V_1, \dots, V_n by applying a shortest path algorithm [Cormen *et al.*, 1990]. Alternatively, if the total number of viewframes N is too great, we use a breadth-first tree search [Cormen *et al.*, 1990] from V_0 to find adjacent viewframes, such that a viewframe cannot appear twice in a path of the tree.

Step 2 If a sequence of *connected* viewframes are not found, stop. Otherwise the robot performs *viewframe centering* and *viewframe back-matching* through V_0, V_1, \dots, V_n . It walks to the common distinct landmark in both V_0 and V_1 .

Step 2.1 If the robot is currently at landmark P of viewframe V_i (i is max), it *landmark circles* V_i , i.e. it walks away from P until P is at its visual range-limit, then it circles around P . During the walk, it tests if current viewframe V_c is adjacent to V_m , i.e. $m \in [i + 1, n]$ and m is max such that $con(V_c, V_m) > 0$; if m is found which means a distinct landmark is found, then it changes the direction and walks to the landmark in *both* V_c and V_m .

Step 2.2 Repeat Step 2.1 until the goal is achieved.

5 LPB Based Navigation Without A Compass For Distinct Landmarks

This navigation algorithm assumes distinct landmarks and no compass and use of LPBs. In [Levitt and Lawton, 1990], the robot uses a global map in its spatial memory to indicate each landmark's estimated direction and distance for path planning. Instead of assuming that the robot knows the estimated direction and distance of each landmark in spatial memory, we assume that the robot only knows the directions of a few selected landmarks called **known landmarks**.

Each pair of the known landmarks forms an **LPB (Landmark-Pair-Boundary)** vector or an **LPB**. LPBs are used to demark visually distinct areas by noting which sides of the LPBs surrounding a region the robot is in. This algorithm uses LPB regions instead of

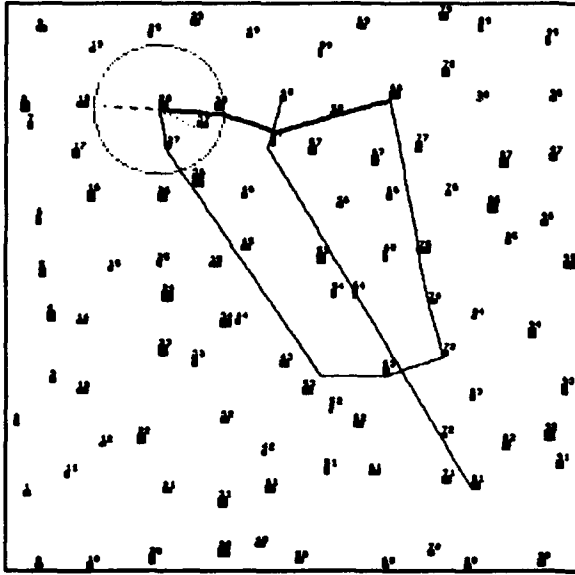


Figure 10: Path Planning from Landmark 68 to 28 (thick path)

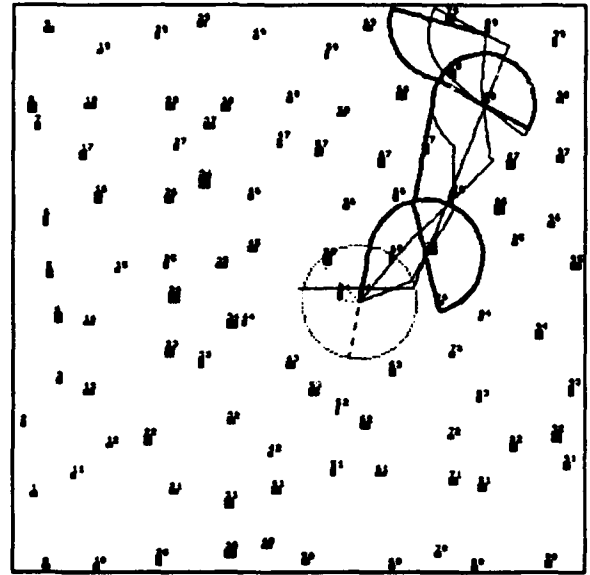


Figure 12: 3rd Time from Landmark 89 to 64 (thick path)

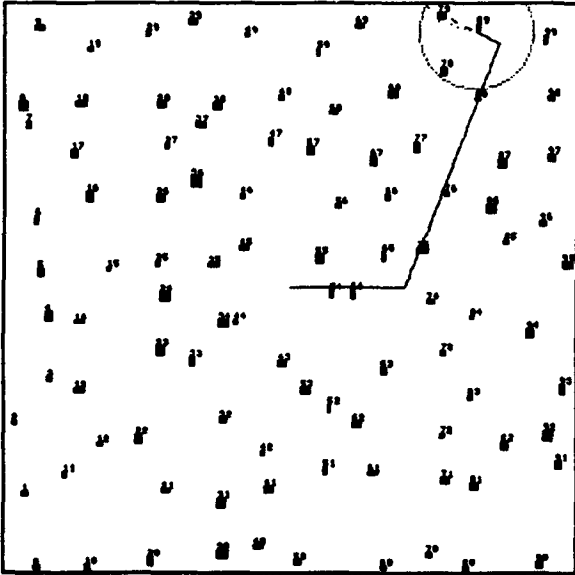


Figure 11: Exploration Path Generated by Random Walk to Landmark 89

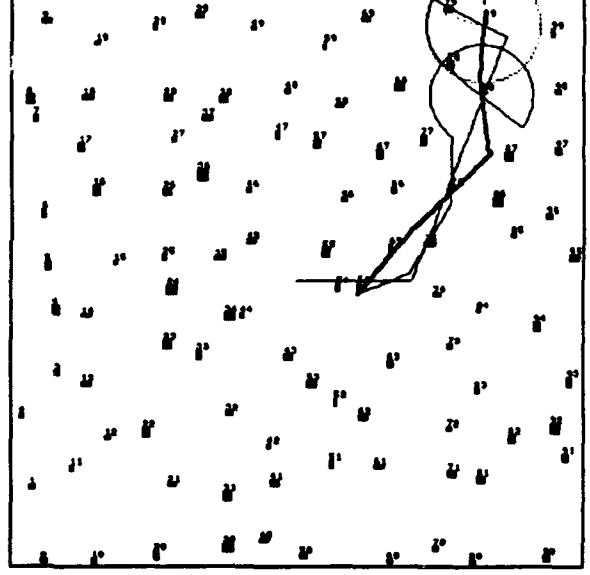


Figure 13: Stable Path (thick path) from Landmark 64 to 89 after 2nd time

viewframes as the basic descriptions of locations. For an LPB vector \vec{l} and a location A , we use $\vec{l}(A)$ to indicate which side of \vec{l} that A is on. $\vec{l}(A)$ has 0, 1, 2 values to distinguish different sides. In Figure 15, known landmarks K_1, K_2 form LPB \vec{l}_{k_1, k_2} . At A , $\alpha_A > \pi$ (from K_1 to K_2 counterclockwise), $\vec{l}_{k_1, k_2}(A) = 1$; At B , $\alpha_B < \pi$, $\vec{l}_{k_1, k_2}(B) = 0$. At C , it's on the LPB, $\vec{l}_{k_1, k_2}(C) = 2$. The two landmarks which define an LPB break the LPB into 3 distinct LPB segments.

Suppose we have n known landmarks forming a total of $(N = \binom{n}{2})$ LPB vectors $\vec{l}_1, \vec{l}_2, \dots, \vec{l}_N$. For a set of LPB vectors $\vec{l}_{k_1}, \vec{l}_{k_2}, \dots, \vec{l}_{k_M}$, we define the LPB projection

for a location A as

$$LPB_prj(A) = \vec{l}_{k_1}(A) \bullet \vec{l}_{k_2}(A) \dots \vec{l}_{k_M}(A) \quad (2)$$

where ... correspond to string concatenation of the values 0, 1, or 2. An LPB region string is the LPB projection using the whole set of LPB vectors determined by all the known landmarks. This creates a net of distinct LPB regions.

Path planning involves finding a sequence of LPB segments from the graph formed by all the LPB segments formed by known landmarks. The robot walks *along* each LPB and tests both sides of it to see if it is adjacent to the goal region. This requires at most $O(n^2)$

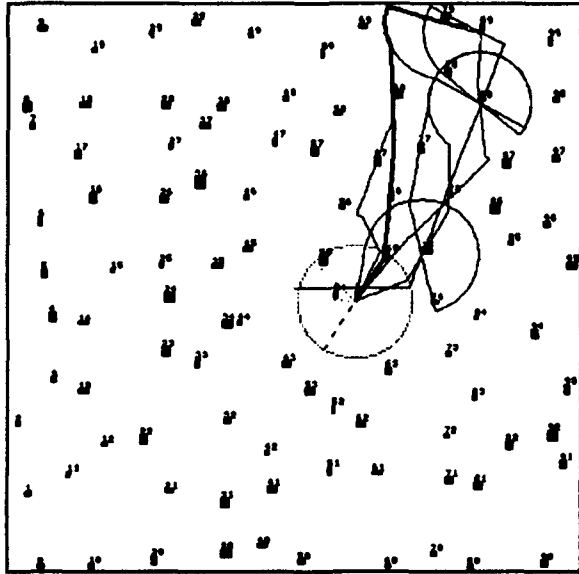


Figure 14: Stable Path (thick path) from Landmark 89 to 64 after 4th time

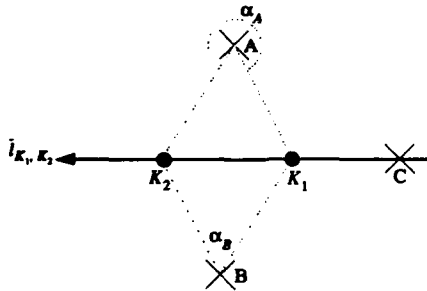


Figure 15: LPB(Landmark-Pair-Boundary) Representation

LPB vectors to be visited. However, we can improve this. There are a total of $(n - 1)$ LPB vectors crossing one known landmark, which will partition the area into at most $2(n - 1)$ distinct sections. Each section is expressed in terms of the LPB projection (onto those LPB vectors) of any location from that section. The basic idea is that the robot goes to the known landmark, walks along parts of 2 LPB segments, which are borders of the section having the same LPB projection (onto those vectors) as that of goal LPB region. The robot then has at most $O(n)$ LPB vectors to visit. In Figure 16, values in parentheses show distinct LPBs projections (onto \vec{l}_{k1} , \vec{l}_{k2} , \vec{l}_{k3}) for sections I through VI; to visit region A (section I), the robot only needs to visit parts $K\vec{l}_{k1}$, $K\vec{l}_{k2}$ of 2 LPB vectors \vec{l}_{k1} and \vec{l}_{k2} .

The algorithm has the following steps:

Goal: A given LPB region and corresponding an LPB region string Lg to go to.

Step 0 If any components of Lg is equal to 2 (it is on an LPB vector), the robot first goes to any known landmark on that LPB. It then walks along it in one direction until the goal region is achieved; if so,

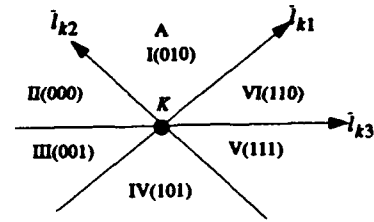


Figure 16: LPB Distinct Section Partitions through One Known Landmark

the algorithm is finished.

Step 1 Initialize segments of each LPB vector as unvisited. Perform **masking** on each known landmark K visited before i.e., mark the segments of LPB vectors crossing K as visited if they are not borders of the section having the same LPB projection (onto these LPB vectors) as that of Lg . Also initialize the stack SP for the known landmarks as empty.

Step 2 Test the stack SP .

If SP is empty,

select any known landmark K which is one end of an unvisited LPB segment, **push**(K) into SP , goto step 2; if K is not found, stop.

Else

$K = \text{pop}(SP)$; if K is not one end of any non-visited LPB segment, go to Step 2.

Step 3 The robot walks to known landmark K , testing whether the goal region is achieved; if so, the algorithm is finished.

Step 4 If K has not been visited before, the robot performs **masking** on K .

Step 5 If K is one end of non-visited LPB segment S , mark S as visited, **push**(K) into SP . Else go to Step 2.

Step 6 The robot walks along segment S , testing whether the goal region Lg is found, until one of the following conditions is satisfied

- if the goal is found, the robot achieves the goal and the algorithm is finished.
- if contradiction to the goal region happens, i.e., originally the robot is on the same side of one LPB as that of the goal, later different; mark visited for the segment which the robot is heading towards, go to Step 2.
- if the robot arrives at another known landmark K_n , **push**(K_n) into SP , go to Step 2.

Figure 17 shows an example of navigation using this algorithm with the viewframes and paths from the previous figures. The known landmarks are circled, the LPB vectors are in dash-dotted line, and the exploration paths Ψ (solid thin lines) are generated in the same manner

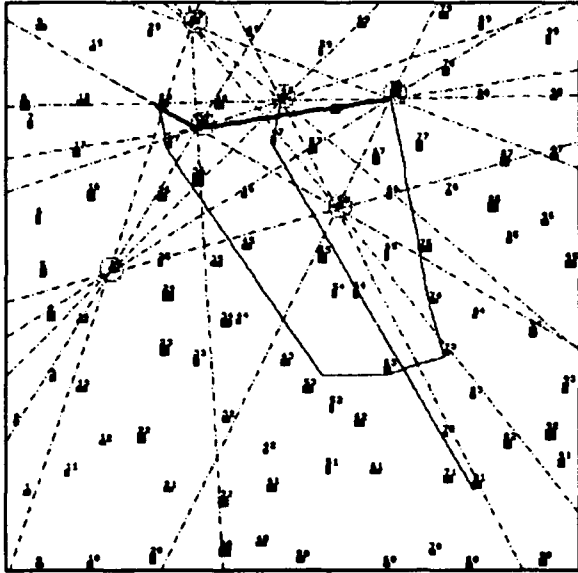


Figure 17: Example of Navigation Using LPBs

as in figures 7, 8, 9. The path determined by robot is shown as a solid thick line from the LPB region near the upper-right corner of the Ψ to the goal region near the upper-left corner of the Ψ . The processing time of this algorithm is $O(n)$ where n is the number of known landmarks. In addition, experiments have shown the algorithm gracefully degrades as the number of known landmarks is decreased.

An interesting finding is, if we apply *masking* on all known landmarks as stated in *step 1* of the algorithm, the LPB candidates (i.e. LPB vectors of which LPB masks are not 111) form a *flow* towards destination region. Figure 18 shows LPBs partition the area into small regions. Figure 19 shows the flow towards the goal region near the Upper-Left Corner of the Ψ of Figure 17.

6 Navigation Not Using a Compass with a Variable Percentage of Distinct Landmarks

We are currently exploring different alternatives for this case. The characteristic behavior appears similar to navigation using a compass with nondistinct landmarks (hugging to previously explored paths without taking shortcuts), except it is much more sensitive to the allowable viewing distance. One approach for this case is to perform navigation using LPBs defined by landmarks with local-ids. A difficulty is that one or both of the landmarks defining an LPB can disappear as the robot walks away from it. So the LPBs connecting viewframes may not be stable. It may be possible to use a measure of reliability of LPBs between viewframes as a criteria for extracting viewframes.

Another approach we are investigating for the case of low percentage of distinct landmarks, involves modifying *viewframe back-matching* in the algorithm from section 3 to satisfy the constraint of not using a compass. In Figure 20, the robot is at A , seeing nondistinct land-

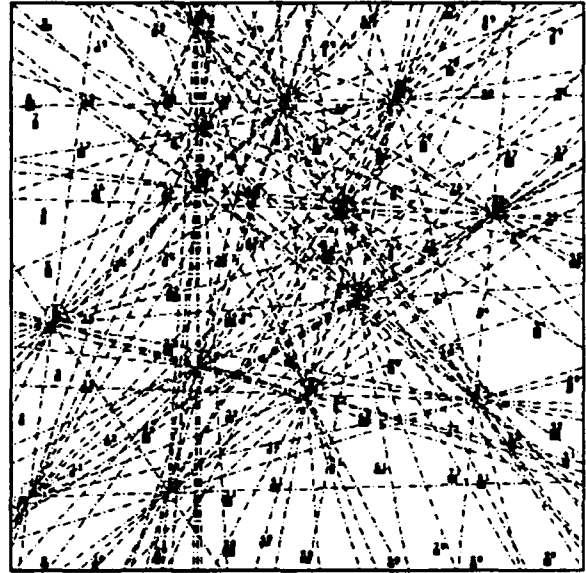


Figure 18: LPBs Partitions the Area into Small Regions

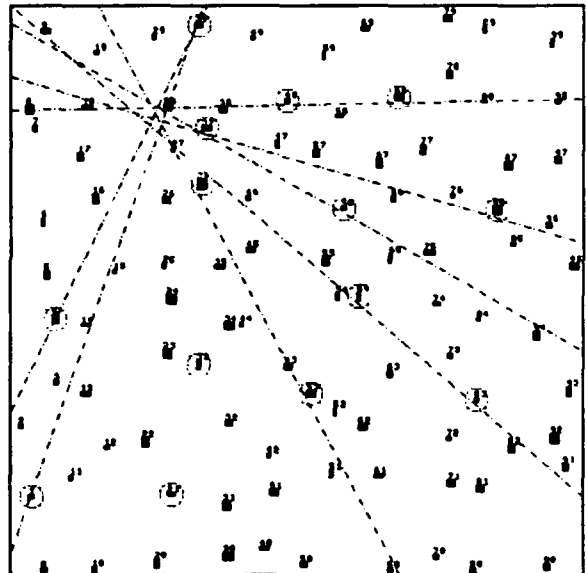


Figure 19: LPB Flow Towards the Goal Region Near the Upper-Left Corner of the Ψ of Figure 17

marks L_1, L_2, L_3 which are also in vf_O . In order to go nearby the center O of vf_O to back-match vf_O , the robot first comes to one of L_1, L_2, L_3 , say L_3 , then it walks along arc L_3BO by maintaining angle $\alpha_2 = \angle L_2OL_3$ walk; and tests if angle $\angle L_1BL_2$ equals α_1 , if so, we conclude the robot is close to O . The robot must always see all 3 landmarks before it comes nearby O (Note angles α_1, α_2 are calculated counter-clock-wise, so there is only one center). This type of walking is used for *viewframe back-matching without a compass*.

The following algorithm is intended for the case of high percentage of nondistinct landmarks without a compass. It has the following steps:

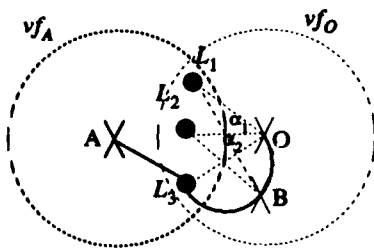


Figure 20: Viewframe Back-Matching Without a Compass

Goal: A landmark with local identifier *lcid* (tied to a specific viewframe) to go to.

Step 0 If *lcid* is in current viewframe, go directly to it and the algorithm terminates.

Step 1 Create a *virtual* viewframe V_n containing only the goal *lcid* with arbitrary orientation. Construct an N by N weight matrix W (N is current total number of viewframes plus one). For each pair of viewframes (V_i, V_j) including the *virtual* viewframe V_n and current starting viewframe V_0 , if they are at least 3 (for $i \neq n$ and $j \neq n$) or 1 (for $i = n$ or $j = n$) landmarks with common local ids in both viewframes, we assign the weight matrix entry $W(i, j) = 1$; otherwise $W(i, j) = \infty$. With the weight matrix, we find a *least* sequence of connected viewframes V_0, V_1, \dots, V_n by applying a shortest path algorithm [Cormen *et al.*, 1990]. Alternatively, if the total number of viewframes N is too great, we use a breadth-first tree search [Cormen *et al.*, 1990] from V_0 to find adjacent viewframes, such that a viewframe cannot appear twice in a path of the tree.

Step 2 If a sequence of *connected* viewframes are not found, stop. Otherwise the robot performs *viewframe centering* and *viewframe back-matching* through V_0, V_1, \dots, V_n . It walks to the landmark with common local identifier in both V_0 and V_1 , where choice of distinct landmark has priority.

Step 2.1 If the robot is currently at landmark P of viewframe V_i (i is max), it finds 2 other landmarks with common local identifiers both in current viewframe V_C and vf_i , and walks towards the center of vf_i by using *viewframe back-matching without compass* explained in Figure 20. During the walk, it tests if current viewframe V_C has at least 3 (1 for $m = n$) landmarks with common local identifiers with vf_m , i.e. $m \in [i + 1, n]$ and m is max; if m is found, which means 3 distinct landmarks are found, then it changes the direction to walk to the landmark in *both* V_C and V_m . Otherwise, it performs *back-matching* to V_i ; if no ambiguity occurs and the best match is found, the robot updates the local identifiers, i.e. it uses local identifiers in V_i to replace corresponding local identifiers with the same orientations in

V_C as well as those local identifiers in $V\text{-DB}$; and then walks to a landmark with common local identifier both in V_i and V_{i+1} .

Step 2.2 Repeat Step 2.1 until the goal is achieved, or failure due to ambiguity.

7 Summary and Future Work

We have described different range-free qualitative navigation algorithms. The data structures we have used, especially for the case of nondistinct landmarks, are compatible with the types of features that could be extracted as landmarks with basic image processing techniques on a robot with a 360 degree field of view. We also have performed experiments to understand path-planning feasibility and efficiency for these algorithms. One measure of path planning efficiency is the ratio of the straight-line distance between two locations compared to the actual distance walked by a robot to go from between the two locations. By this measure of efficiency, the compass-based algorithms improve if number of viewframes, visual range, and number of distinct landmarks increases. The non-compass-based algorithms also depend on allowable visual range. The efficiency of the LPB, non-compass-based algorithms increase as the number of known landmarks increases.

Our current work is focusing on navigation using LPBs formed from nondistinct landmarks, viewframe filtering techniques, and different approaches to organizing spatial memory, such as a hierarchical representation of viewframes, along the lines discussed in [Kuipers, 1978].

References

- [Cormen *et al.*, 1990] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
- [Gallistel, 1990] C. R. Gallistel. *The Organization of Learning*. The MIT Press, 1990.
- [Kuipers and Byun, 1987] B. J. Kuipers and Y. T. Byun. A qualitative approach to robot exploration and map-learning. In *Proceedings of the Workshop on Spatial Reasoning and Multi-Sensor Fusion*, Los Altos, CA, 1987.
- [Kuipers, 1978] B. J. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129-153, 1978.
- [Levitt and Lawton, 1990] Todd S. Levitt and Daryl T. Lawton. Qualitative navigation for mobile robots. *Artificial Intelligence*, 44:305-360, 1990.

Interactive Model-Based Vehicle Tracking*

Daryl T. Lawton, Warren F. Gardner, and Jun-Hoy Kim

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

Abstract

We describe an architecture for an interactive model based vision system and its application for vehicle tracking. A human specifies a limited amount of information which establishes a context for autonomous interpretation of images obtained by a telerobot. Object models are described by constraints specifying necessary geometrical properties and relationships between objects. The use of constraints allows for flexible object instantiation. A user can indicate a vehicle and this directs perceptual processing routines to determine the corresponding local surface orientation and roads, or he can instantiate a road segment to direct the extraction and tracking of vehicles. We conclude with a processing example based upon implemented components and a brief discussion of future work.

1 Introduction

Efforts to develop intelligent and autonomous systems for operation in complex, natural domains have been largely unsuccessful to date, in spite of continued advances in the underlying technologies. There remain unresolved and fundamental difficulties in terms of the necessary computational power, the required complexity of perceptual systems which can operate in outdoor environments, and the corresponding complexity of planning and reasoning systems. A recent framework addresses many of these problems by stressing the importance of telerobotic and interactive systems. This is a realistic approach to fielding advanced technology in the short term, and also provides a long term framework for developing autonomous systems. An interactive, semi-autonomous system can significantly amplify the capabilities of a human, and also yields an evolutionary approach as autonomous system capabilities are developed and begin to replace human controlled functions.

*This research has been supported by the U. S. Army Human Engineering Laboratory and the Advanced Research Projects Agency of the Department of Defense (monitored by the U. S. Army Topographic Engineering Center under contract No. DACA76-92-C-0016)

The approach described here is to develop a model-based vision system that a human can interactively control. The human uses this to rapidly interpret sensory information from a potentially distributed team of telerobots. The resulting interpretation is a model of the world that the telerobots can refine, use to control their behavior, or report back to a human. In this way, the human directs the telerobots by initializing and constraining their processing. Communication between the robot and the human can then take place in the context of a shared model of the world which makes possible infrequent, semantically meaningful, and low bandwidth communication.

The particular system we present is for tracking vehicles in outdoor scenes. A human can manipulate models of objects such as terrain surface patches, roads, and vehicles to interpret imagery from a telerobot. Once an interpretation is in place, the telerobot can autonomously refine and extend the interpretations, detect and track vehicles, and report back to a human about unusual occurrences or behavior that cannot be accounted for. For example, a human will indicate that a particular area is a road. The vision system will then track movement along the road and fit a constraint-based description of a vehicle to this movement. The system could determine that a vehicle has just gone off the road (or that it is behaving inconsistently with respect to the model of a vehicle).

We begin by reviewing the basic architecture guiding the development of the interactive model based vision system, and then detail some of its components involving object models and perceptual processing that have been implemented.

2 System Architecture

The underlying architecture is shown in Figure 1. It is built around three major data bases that a human can access and manipulate through a user interface. The basic task of the human is to access models of the various types of objects stored in the **Object Model Data Base** along with information describing maps, landmarks, and previous interpretations in the **Long Term Data Base** to build an interpretation of the current scene which is stored in the **World Model Data Base**. For example, the human is presented with images from cameras on the telerobot. He can use a priori

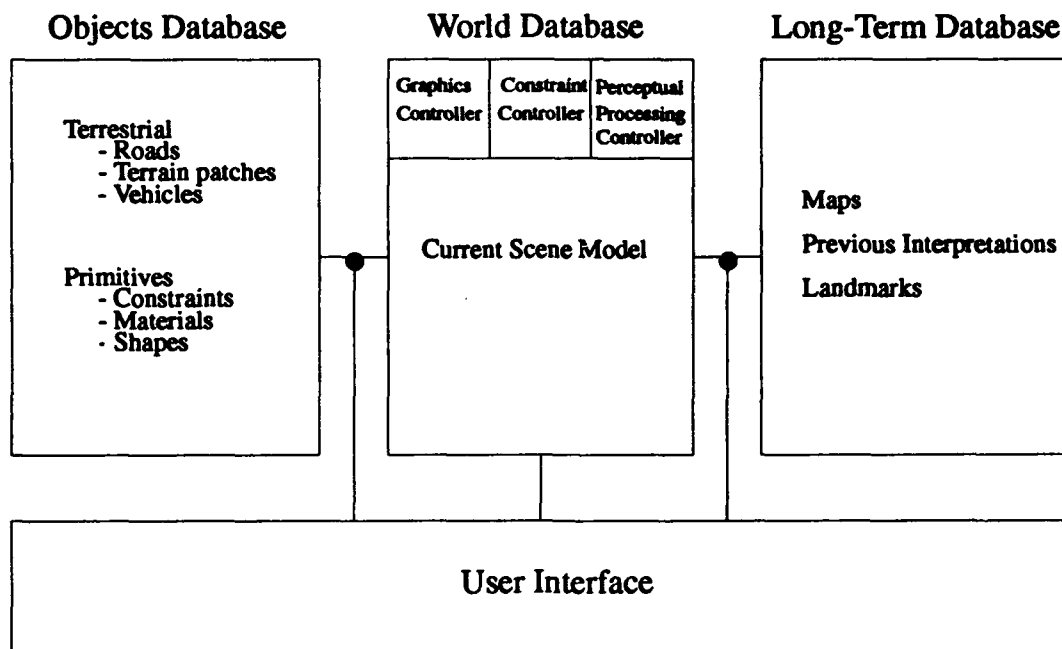


Figure 1: System architecture

maps to align landmarks and terrain features from these maps with the images. He can also access the three-dimensional and physically based models of objects and position them with respect to the world model. As he does this, the models are projected back against the images obtained from the telerobots for interactive control and to initiate processing.

The **Object Model Data Base** contains generic models of objects, relationships, and events for terrestrial scenes. This involves objects such as terrain patches, roads, vehicles, and gravity. We distinguish between two different types of objects: **Primitives** which correspond to basic entities and relationships used to describe and represent **Terrestrial Objects** which correspond to the conventional objects found in the world such as roads and cars. Primitive Objects describe characteristics such as shape constraints, material composition, and relationships between parts. The representation of objects for an interactive vision system is more complex, though related in many ways, to those used in CAD/CAM and geometric modeling packages, because they will be manipulated for autonomous processing and reasoning. Thus, in addition to describing shape, the model of a car needs to include that a car is acted on by gravity and will have a preferred type of orientation and attachment with respect to the ground surface. Object models are described by sets of constraints [Borning, 1981; Lawton, 1980; Leler, 1988; Mundy *et al.*, 1989] which must be satisfied. A simple constraint is that the value of some parameter associated with an object model is bounded. More complicated constraints deal with relations between objects. The human will in general specify a limited amount of information

for an object, and the system will use the constraints and associated processing actions to then refine the instantiation of an object.

The **World Model Data Base** describes the three dimensional world of objects and situations surrounding the telerobots. It is initially formed by the human accessing models in the object data base and instantiating them. There are three types of controllers associated with the World Model Data Base. The **Constraint Controller** checks for consistency in the world model. The constraint controller uses the constraints which define an object or relationship to refine an instantiation or to find a violation or inconsistency and ask the human for help. The **Perceptual Processing Controller** deals with the extraction of information from images and sensors on the telerobot. The constraints in an object model specify the types of processing that are necessary to obtain this information. When the human indicates that a road is located somewhere, this constrains the type of tracking and feature extraction processes that are used. The corresponding image areas are isolated and the type of segmentation or tracking procedure corresponding to the material class and distance of the object is applied. The **Graphics Controller** deals with interactive scene measurements and the presentation of the world model to the user. Thus when he accesses a model of a vehicle, he is presented with a cartoonish three-dimensional vehicle template which is back projected onto the image being interpreted.

The user interface is currently based upon windows for displaying imagery and graphical overlays, and text-based browsers for inspecting entities in the data base in detail. This basic level of interface can be quite te-

dious to work with and its future role will be to serve as a debugging tool. An intermediate, near-term system interface will use a more natural set of tools such as three-dimensional hand/finger position sensors and voice input. Using these, the human will actually have a sense of reaching into the data base of models, grabbing something, and then placing it into the world model. In the eventual system, the world model and the sensor input from the different telerobots could be presented to the human as a virtual reality in which the human can be embedded in the world model itself.

3 Object Models

We have currently developed models for objects corresponding to gravity, the immediate ground plane surrounding the camera from which an image is obtained, terrain patches, and a generic vehicle along with constraints describing relations for attachment, alignment, and coincidence. There is a simple mechanism to invoke the instantiation of models based upon other models that have been instantiated and the results of perceptual processing. A more powerful constraint propagation mechanism would determine consistency of relationships between objects in the world model data base.

Each object model has a two-dimensional image registered display that can be interactively manipulated. When this is instantiated it sets up perspective constraints with respect to the three dimensional object models which will have unspecified parameter values. For example, gravity appears as a two-dimensional vector field that can be interactively aligned with image features. We use different types of road models for two and three dimensions. The two dimensional road model is a sequence of connected parallel line segments for the road boundaries and/or the center-line of the road. This is used to indicate and mask images areas which are adjacent to the road. The three dimensional road model is a connected sequence of segments with three-dimensional coordinates and associated road width information with constraints on allowable orientations with respect to gravity and adjacent terrain patches. The end-points of the linear segments in the two-dimensional display of a road model can constraint the three dimensional parameters for positioning the road model. Different material properties can be associated with the roads, but this currently isn't used by the segmentation and feature extraction procedure.

The generic vehicle model is an oriented box with an indication of where the track/wheel area of the vehicle is, where the engine is positioned, and where the cab area is. The scale and relative position of these is parameterized and can be specialized for different types of vehicles. There are scale and orientation constraints on all of these components as well as for relative position to ground surfaces and gravity (see Figure 2).

4 Perceptual Processing

Image processing and tracking procedures are organized in terms of the type of information they depend on and can extract. One type of tracker depends on a two or

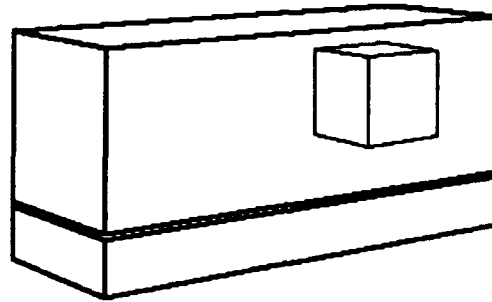


Figure 2: Perspective view of the three-dimensional vehicle model

three dimensional road model and can yield information to instantiate a vehicle model. An instantiated vehicle model constrains the extraction of features. These features satisfy the requirements of another type of tracker that can determine a scaled three dimensional trajectory for extracted image points. The information determined by this tracker can in turn be used to determine a three dimensional road model, and also refine the attributes of an instantiated vehicle model. As a result, the flow of information and processing varies based upon the state of the current interpretation. The current processing routines consists of three types of trackers and restricted segmentation and interest operators which are applied when a vehicle model is instantiated.

4.1 Difference Tracker

The difference tracker operates with respect to an instantiated two or three dimensional road model. It determines regions above the indicated road areas which are changing overtime and are also moving in a consistent direction (not necessarily along the road). It determines information to instantiate a vehicle model by finding the front and back (or only the back or the front) of a vehicle. If a three dimensional road model has been instantiated, it can further constrain the dimensions of the generic vehicle model instantiation. It also restricts the extraction of features for the local translational tracker (Section 4.2) which can in turn recover the direction of motion of the vehicle, whether it is turning, and the corresponding direction of motion relative to the road.

The first step in the difference tracker is to reduce the image noise by convolving consecutive images in a motion sequence with a low-pass filter. If no models are present, the entire image must be convolved with this filter. However, given a two-dimensional road model, the filter is only convolved with pixels that are above the road. The road model shown in Figure 3 is used to constrain the smoothing process.

Once the images have been smoothed, the algorithm begins to search for areas of motion that lie near the road. This is accomplished through image subtraction. Pixels from temporally consecutive images that are situated near the road model are subtracted. If the result of this subtraction is greater than a threshold, the environmental object corresponding to this pixel position is

assumed to have undergone motion. This pixel is marked as a motion pixel, and a region growing process begins.

An object traveling along the road may extend some distance from the road (i.e. the object could be very close to the camera, in which case it would appear to be quite large). The search for all areas of motion associated with an object is accomplished through region growing. Once a pixel near the road has been identified as a motion pixel, its neighbors are also examined using the subtraction technique discussed above. If any of the neighboring pixels contain motion, their neighbors are also examined. This recursive procedure continues until no more motion pixels can be found. An example of this extraction of the areas of motion is shown in Figure 4. Once the areas containing motion have been identified, the centroid of these areas is located. Over time a two dimensional trajectory can be constructed.

4.2 Local Translation Tracker

Moving vehicles can often be treated as rigid objects which are translating over short periods of time. For example, as a vehicle goes around a curve, because of turning radii constraints, the axis of rotation is often far away from the vehicle itself and the vehicle motion can be treated as a sequence of small translations corresponding to tangents of the curve of motion. The local translation based tracker determines the direction of motion of a set of extracted image points over time, and fits their motion to an estimate of the current direction of motion of the corresponding vehicle in three dimensions. Essentially, it determines the direction of motion of a set of environmental points over time. The effect of this tracker can be visualized as a unit sphere with an axis corresponding to the current direction of motion. As the vehicle and the corresponding set of points move, the position of the axis changes with respect to the sphere. This processing works well with temporal filters since there are constraints on how quickly a vehicle can change its direction of motion. This can also be used to determine if a vehicle is rotating with respect to an axis contained within the vehicle. This is indicated by areas of the image which show differences over time, but for which no clear axis of translation can be determined.

This tracking algorithm is based on the strong geometric constraints on image motion in the case of translational motion (radial motion of image features from a focus of expansion, determined by the intersection of the direction of translation with the imaging surface) [Lawton, 1982]. The algorithm evaluates an error measure which associates with a potential axis of translation, the quality of feature displacements along the corresponding radial flow paths. This error measure is evaluated by searching over a unit sphere which describes all potential directions of translation. It is possible to determine the direction of translation to within a few degrees in small image areas, using only a few features.

If there is an instantiated three-dimensional road model and a rough estimate of the position of the vehicle along the road has been established, the tangent information associated with the road model can be used to initialize the search for the axis of translation. If there

is an instantiated vehicle model, it restricts the features that the local translational tracker uses.

4.2.1 Feature Extraction

The local translation-based tracker requires features which can be matched in successive images. The type of features we use are conventional masks of image pixels, extracted from distinct areas of the image. In the examples shown in this paper, the masks are 5x5 pixel arrays. We have used normalized correlation [Ballard and Brown, 1982] to determine similarity of extracted features. This is used in measuring feature distinctiveness and for evaluating the matches of extracted features along the radial flow determined by a possible axis of translation. Since the radial flow lines do not necessarily pass through the center of the image pixel arrays, we use bilinear interpolation for matching features.

The distinctiveness of a feature is 1 minus the best correlation value obtained when the feature is correlated with its immediately neighboring areas. Good features are selected by finding the local maxima in the values of the distinctiveness measure over an image. We constrain the neighborhoods over which the features are selected to areas that contain large intensity discontinuities, determined by extracting zero-crossings. The area of feature extraction is further constrained by the output of the difference tracker or an instantiated vehicle and road model. The distinctiveness measure is then applied only to these restricted areas in an image. This generally results in the extraction of areas of high curvature along the zero-crossing contours. In addition, as a vehicle is tracked over a sequence of images, this processing is continually reapplied to find features in addition to those that have matched successfully. These can correspond to new features due to occlusions or changes in observable detail as a vehicle moves in depth.

4.2.2 Determining the Direction of Translation

Features in image sequences will move along radial lines defined by the focus of expansion (FOE) during translational motion. The FOE is determined by intersecting the direction of translation with the imaging surface (where the direction of translation emanates from the focal point of the camera). Using this geometric relationship, the displacement paths of all image features can be determined for a potential direction of translation. To evaluate a potential direction of translation, we search for each feature along the appropriate image displacement paths. The error measure used to evaluate this potential direction of translation is determined by summing the best matches for each of the features.

To search for the direction of translation we use a unit sphere centered at the focal point of the camera. Any vector which has its initial point at the camera's focal point and its terminal point resting on the surface of the sphere is a potential direction of translation. The search procedure is defined with respect to this sphere instead of the potential positions of the FOE in the image plane. This is because the sphere is a bounded surface which makes uniform global sampling of the error measure feasible. When the image plane is used directly, the

resolution in the position of the translational direction varies.

The initial search process consists of two phases: an initial global sampling of the sphere, followed by a local search for the maximum value. The local search begins at the position of the maximum value as determined by the global sampling. The local search process recursively searches the area of the current maximum. The step size of the local search processes is reduced until it is at the desired resolution for the determination of the direction of translation. Figure 6 shows a sequence of tessellated spheres along with their potential directions of translation. Once a direction of motion has been established, it will tend to change smoothly and we can then use gradient based techniques to track the axis of translation for successive images. In addition, if there is an oriented vehicle model or a road model segment, the search for the translational axis is constrained to limited areas of the sphere.

4.3 Planar Tracker

Often the motion of a vehicle is restricted to a plane determined by the local road or surface orientation. In this case, the geometry of planar perspective makes it possible to associate three dimensional information with extracted image features if they are contained in the area of the planar patch. In addition, the directions of motion are constrained to be parallel to this plane, so the possible directions of motion for the local translation-based tracker are restricted to a circle on the unit sphere whose orientation is parallel to the plane of motion. This simplifies initialization and also tracking of the axis of translation over time.

There is another useful constraint associated with planar motion that may not be immediately apparent. In this case, an environmental displacement vector v must be perpendicular to the normal of the plane of motion. v also lies in the plane determined by its corresponding image displacement and the focal point of the camera. The direction of environmental motion can be determined by intersecting these planes. This is useful for tracking planar motion without the constraints supplied by a road model.

4.4 Feature Extraction from a Model

When the vehicle model is instantiated it constrains segmentation and feature extraction procedures to a limited image area. In addition to the feature and zero-crossing extraction described above, we use histogram based segmentation to determine potential vehicle features.

An instantiated vehicle model can also constrain the places to search for detailed features corresponding to portions of the vehicle which can be tracked. A particular problem we have found is that it is necessary to have a large image area to get clear views of the features to be matched to the model. Images of the vehicle will need to be larger to begin finding detailed features such as headlights, bumpers, and so forth. Such images could perhaps be obtained by using one of the trackers to direct a zoom camera to follow a moving vehicle. Currently we use the interest operator described for the transla-

tional tracker to match extracted features to a vehicle model for each successive image. If extracted features are near previously extracted features that have successfully matched they are discarded. Otherwise, they are associated with the instantiated vehicle model.

5 User Interface and Model Instantiation

An important facility in the user interface is a conventional depth buffer used for hidden surface removal which has been modified to have pointers, ordered by depth, to all the objects in the world that project onto a given pixel in the image. Thus, when the human "touches" a pixel in the image from the telerobot, he can access all the objects in the world model that project onto that pixel. We call this an augmented depth buffer.

The user interface enables the human to place objects into the world model in several ways. He can access the objects and manipulate them via their three dimensional attributes with respect to a coordinate system linked to the world model. This looks like back projecting a three-dimensional cartoon of the object onto the image. When it has been positioned as desired, the different components of the object can be placed in the augmented depth buffer associated with the image. In this way, the projected attributes of the instantiated object can access the actual image or the results of image processing routines. The user can burn-in attributes when he instantiates an object. Burning-in means that the attributes can not be changed. This often involves constraining a particular feature to lie along a given ray of projection. Another technique is for the user to directly draw the specified object on the sensory input and then indicate its attributes. An example of this is interactively segmenting an image into different types of terrain patches and pointing out that different edges correspond to terrain feature discontinuity.

6 Processing Example

An example of this processing is shown in Figures 3-6. Figure 4 shows a sequence of images obtained with a video camera viewing a road scene. In Figure 3 a human has interactively positioned a generic vehicle model with respect to the road and has begun to "drive" the model vehicle through three dimensions while using the back-projection of the vehicle as a three dimensional cursor. Note the center segments of the road being laid down behind the vehicle. This establishes a two-dimensional road mask and also an initial set of connected three-dimensional road segments to constrain later processing. Figure 4 shows connected regions of image differences moving in a consistent direction with respect to the user instantiated road model. These correspond to the front and back of a vehicle. Since orientation is known along the road and the road model has been scaled relative to the generic road model, it is possible to use these areas to instantiate a three dimensional vehicle model. Figure 5 shows interesting points which have been extracted in the corresponding areas determined by the vehicle model. These features are then used by the translation

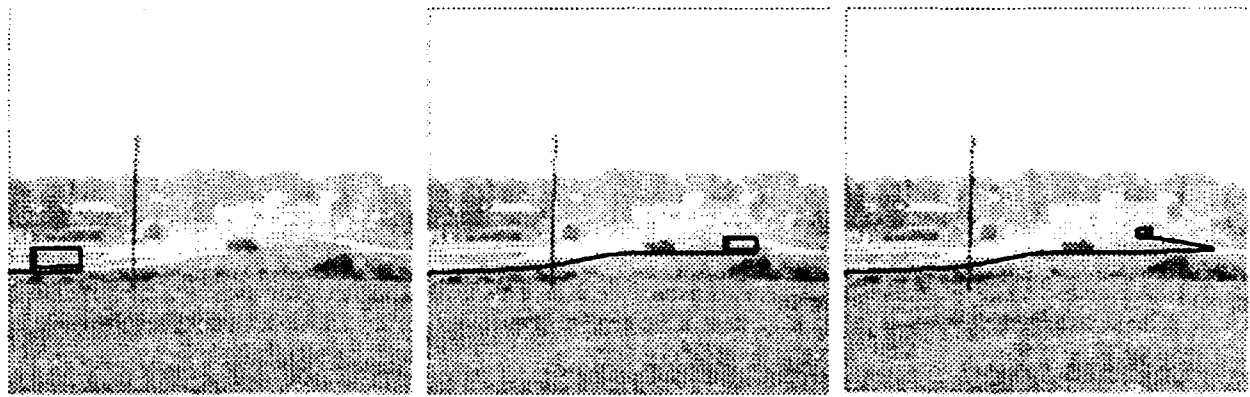


Figure 3: Interactively driving the vehicle to form the road model

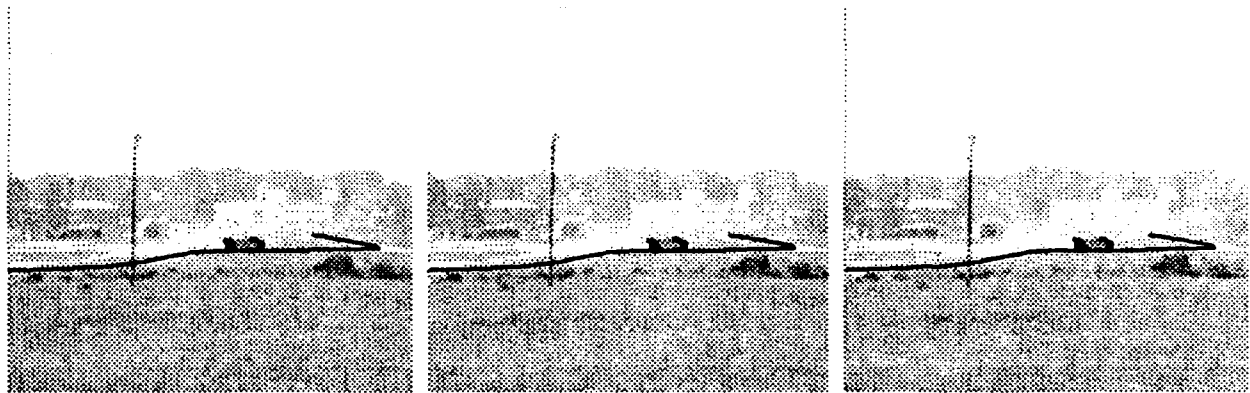


Figure 4: Areas of motion and vehicle position found through differencing

tracker to refine the estimate of vehicle and road orientation. The determined successive directions of translation are shown in Figure 6. More and more features are associated with the vehicle model over time.

7 Future work

Our current work involves:

- Extending the number and complexity of the models that are used along with the a more general constraint propagation mechanism.
- Extending the user interface to use a wide range of interactive devices such as a data glove and other three-dimensional positioning devices.
- Integrating the local translational tracker with a Kalman Filter for processing over time.
- Using multiple cameras from different points of view with respect to the same scene.

References

- [Ballard and Brown, 1982] Dana H. Ballard and Christopher M. Brown. *Computer Vision*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1982.
- [Borning, 1981] A. Borning. The programming language aspects of thinglab, a constraint oriented simulation laboratory. *ACM Transactions on Programming Languages and Systems*, 3(4):353ff, 1981.
- [Lawton, 1980] Daryl T. Lawton. Constraint-based inference from image motion. In *Proceedings of AAAI-80*, 1980. Stanford, CA.
- [Lawton, 1982] Daryl T. Lawton. Processing translational motion sequences. *Computer Vision, Graphics, and Image Processing*, 22:116-144, 1982.
- [Leler, 1988] W. Leler. *Constraint Programming Languages: Their Specification and Generation*. Addison-Wesley, Reading, MA, 1988.
- [Mundy et al., 1989] J. Mundy, P. Vrobel, and R. Joynton. Constraint-based modeling. In *Proceedings of the Image Understanding Workshop*, 1989. Stanford, CA.

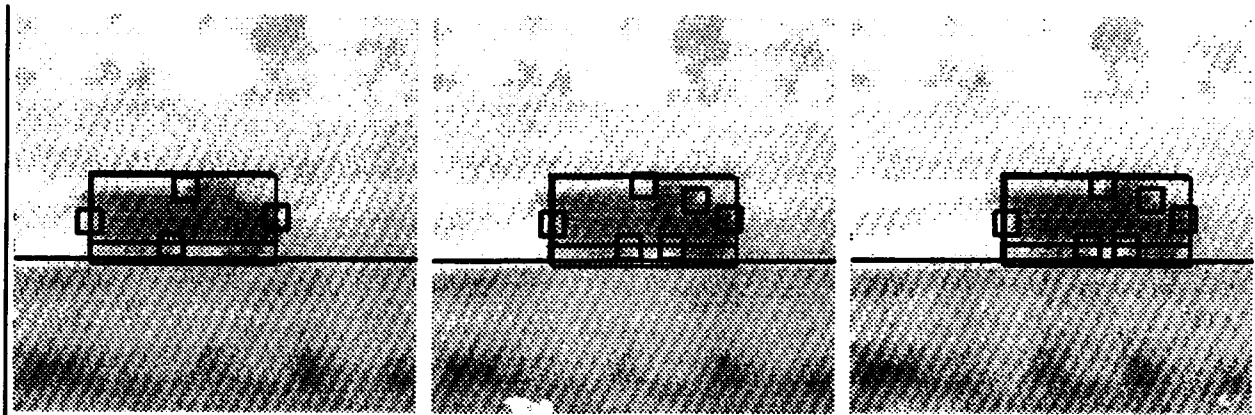


Figure 5: Extracted features with a superimposed three-dimensional vehicle model

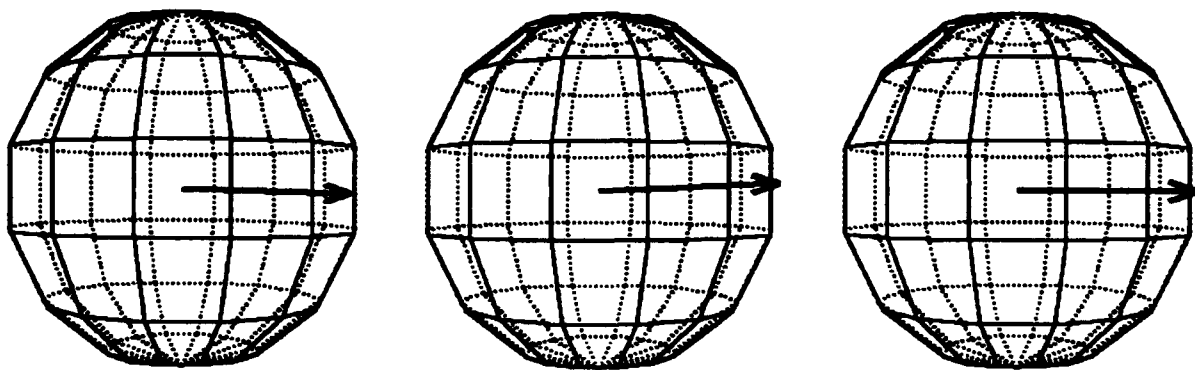


Figure 6: Translational motion spheres corresponding to the image sequence

Qualitative Environmental Navigation

Il-Pyung Park and John R. Kender
Department of Computer Science
Columbia University
New York, NY 10027

Abstract

In this paper we describe a purely topological method for navigation in a large unstructured environment that contains featureless objects, using qualitative non-metric information such as "isolated" landmarks and "trajectories", which we define. The map-maker and the navigator are implemented using an IBM 7575 SCARA robot arm, PIPE, and two cameras. The navigational environment consists of a flat plane with identical spherical objects populated randomly but densely on it. First, the map-maker model observes the environment, and given a starting position and a goal position, it generates a "custom map" that describes in a non-metric language how to get from the starting position to the goal position efficiently and reliably. The accuracy and the cost of the directional instructions are analyzed, then demonstrated by the navigator by following the commands in the custom map. Several non-intuitive and ill-specified aspects of navigation in this manner are then discussed.

1 Introduction

Navigation in a large unstructured environment requires different information and tools than that of navigation in a structured small environment. To guide a navigator in a such environment, the direction giver has to produce a set of directional instructions that contains not only sufficient information for accurate navigation, but also has to make sure that the given set of directional instructions is not overburdening the navigator with too much information [Streeter *et al.*, 1985]. In this paper, we explore the effectiveness of navigation using "isolated" landmarks and "trajectories", both of which make use of

qualitative information rather than quantitative one.

2 Definitions

2.1 The world, the map-maker, and the navigator

There are two major modules in this project, namely, the map-maker and the navigator. They both operate on the navigational world.

The navigable world

The navigational environment that we are interested in is a three dimensional world, although the current implementation of the navigator (due to its restriction of degrees of freedom) makes the effective environment two dimensional. The navigational terrain itself is a flat surface that is visually uniform all over. Objects that will be scattered over this flat surface are spherical objects, such as marbles, uniform in size. There is no restrictions on where the objects are placed, except that no objects are allowed to be placed on top of another. However, the two assumptions of the objects - that they are uniform in size, and that they are placed randomly - emphasize the spatial and topological problems of doing vision and navigation "in the large". That is, firstly, an object can no longer be described by its intrinsic attributes, such as shape, size, or color. Therefore, in order to describe an object, the geometrical relationship of the target object to its neighboring objects must be considered. Secondly, the directions of the movements of the navigator have no external reference to rely on. Our goal is to be able to throw a number of marbles on a table and have our robot successfully navigate through this random world. To a large extent, we have succeeded, but by using disks.

	Map-maker	Navigator
Objective	Generate a custom map	Navigate using a custom map
Visibility	Infinite	Limited to current view window
Metric ability	Yes	Only within current view window
Intelligence	Omniscient	Limited to interpreting the custom map
Memory	Large	None
Computing power	Fast	Slow

Table 1: Capabilities and limits of the map-maker and the navigator

The map-maker

The purpose of the map-maker is to generate a "custom map", consisting of directional instructions, so that the navigator will be able to find its way to the destination by executing each of the instructions in a sequence. The map-maker is assumed to be omniscient and error free. It sees the whole environment and knows the exact position of each objects that exists. The map-maker also knows the capabilities and the limits of the navigator. Therefore, the custom map contains directional instructions that the navigator can handle. The communication between the map-maker and the navigator is done off-line (one-way, one-time). This means that the custom map is given to the navigator in the beginning of the journey and the navigator's only source of directional information is the custom map.

The navigator

The navigator's capabilities are much more limited. Its view window size is very small relative to the environment. It has limited metric measurement capabilities only within its current view window. However, the accuracy of the its metric measurements is assumed to be low. For example, the navigator can move the window position so that a visible reference object is positioned near one of the four corners of the view window, but it does not have the accuracy to pinpoint the coordinates of a visible object. Therefore, it is not possible for the map-maker to give, to the navigator, the (x, y) coordinate of a landmark as part of the directional instructions. The navigator is not intelligent enough to decide on its own what it should do, and is thus totally dependent on the custom map that it is given. It does not have any memory. Therefore, the custom map can be considered as the navigator's intelligence. Table 1 summarizes the assumptions that we make on the map-maker and the

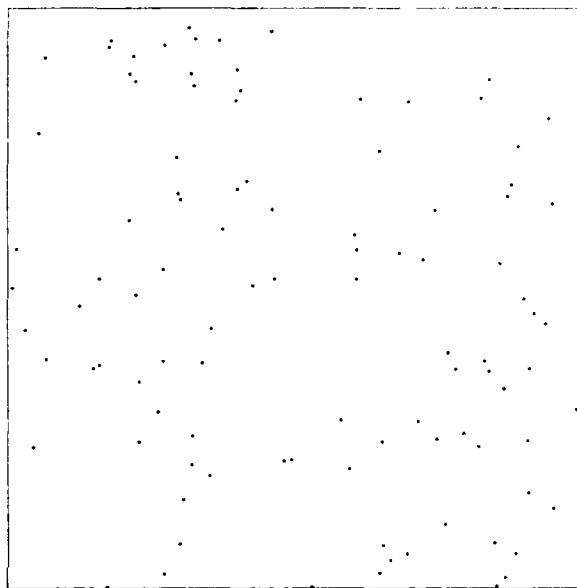


Figure 1: Randomly populated environment of disks on a table

navigator.

2.2 Parkways and trajectories

First, the map-maker captures the entire world by taking the image of the world from a vertically high location. The positions of the populated objects are recorded. Figure 1 shows an example of a randomly populated world with 100 objects. The map-maker is assumed to know in advance the capabilities of the navigator, such as view window size and degrees of freedom. The map-maker further abstracts the world into a graph data structure, with vertices and edges. There are many ways to decide whether or not two vertices (objects) in this graph are connected by an edge. One way is to define that two objects are "connected" if the two objects can be viewed in the same view window of the navigator. By applying the connected component algorithm using

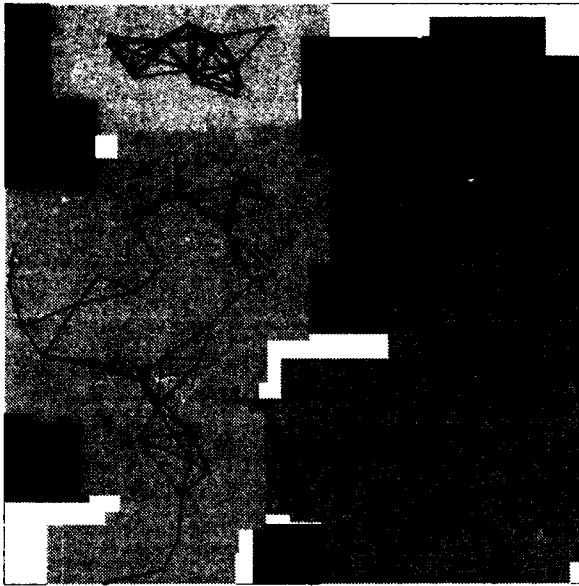


Figure 2: Parkways, the paths that allow step by step movements of the navigator

the above definition of "connectedness", we can then generate discrete sets of connected components. We define a parkway to be such a connected component. Figure 2 shows parkways formed on a world of 100 objects using a window size 10×10 in a world of size 100×100 . The relative size of the view window to the world is shown in the lower left corner of the figure. The arcs between points indicate that the objects are connected, that is, both are simultaneously visible in same window. To find the shortest path between two objects in the same parkway, we apply Dijkstra's shortest path algorithm.

To find a path between two objects that are in mutually separate parkways, we need to be able to devise methods to transfer between parkways. At some point of the traversal, the navigator has to leave a parkway and get to the other parkway without getting lost. In our method, we "slide" the view window in the direction formed by two objects within the view window until a new object is reached. The inter-parkway paths generated by this method are defined as "trajectories". The overall shortest path is then the appropriate combination of parkway paths and the trajectory paths. Figure 3 shows an example of trajectories computed on our random world.

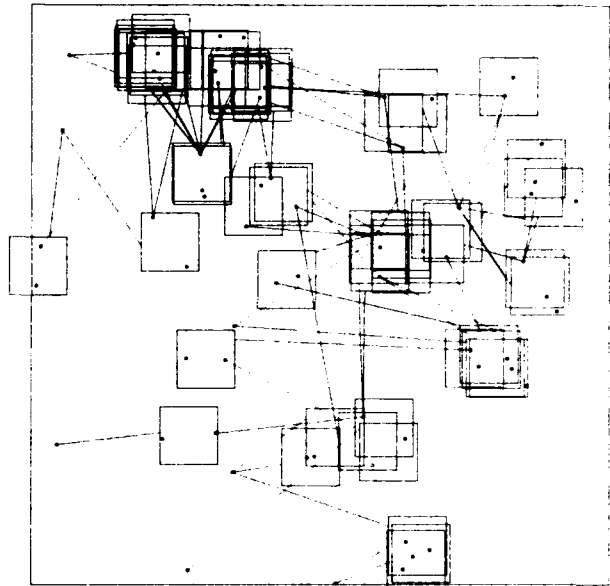


Figure 3: Trajectories

The small boxes are the navigators view windows. The straight lines are the trajectories of the windows in the direction of the sliding movement. One end of a line is attached to the window at the position to which the new object is expected to appear. The other end of the line is the object that this sliding window is seeking.

2.3 Description language

The map-maker needs to generate a custom map that describes how the navigator may follow the landmarks along the computed shortest path. In this section, we explore the issues in designing the language for the custom map. At any given instant, the navigator will have only a small portion of the world at its view, which may contain several objects. The map-maker has to be able to describe what the navigator sees in order for the navigator to be able to distinguish a particular object to use as the next reference point. This is a hard problem because of our assumption of the navigable environment which is comprised of point-like objects that are randomly placed. If the robot has infinitesimal accuracy, infinite memory and extremely fast processor, it could keep the bitmap image of each possible view, or at least the coordinate of each landmark within each possible window. But realistically, we need some invariants, such as colors or shapes, to de-

scribe the immediate environment. Since our world is a monochrome point-like world, we need a qualitative language that can describe the geometrical relations of the visible objects. The level of detail in the description process would depend on the intelligence and the mobile ability of the navigator; our exploration here deliberately emphasizes the topological aspects of navigation.

Some vocabulary follows: An object is defined to be "obvious", if it is the only other object that is visible except for the reference landmark. Two or more objects are defined to be "confusable" if it does not matter which object is to be chosen as the next reference point. The term "new" objects refers to the objects that newly came into the view window as a result of robot's movement. The term "isolated" point refers to a single isolated object as a result of applying a clustering algorithm. The term "isolated pair" refers to a single pair of isolated objects as a result of a clustering algorithm. Figure 4 illustrates these methods. All have been or are being implemented, but we will only talk about one of these vocabulary terms in this paper. Figure 5 shows an example of a navigation using the description language. The custom map entries corresponding to each movement is shown to the right of the figure. Each entry is of the form $(D, [T], M)$, where D is a description language vocabulary that identifies a landmark, $[T]$ is an optional term that indicates trajectory, and M is the corner designator to indicate which of the 4 corners (SE, SW, NE, NW) the chosen landmark is to be positioned. The symbol * is the wildcard that matches any D or M .

3 Isolated landmark following and trajectory traversal

In this paper, we present one navigational technique using the isolated point descriptor and the trajectory method. Parkway traversal is done by following isolated landmarks solely. Parkway crossing is done by using the trajectory method solely. As stated, other descriptors and methods have been or are being developed.[Park, 1993]

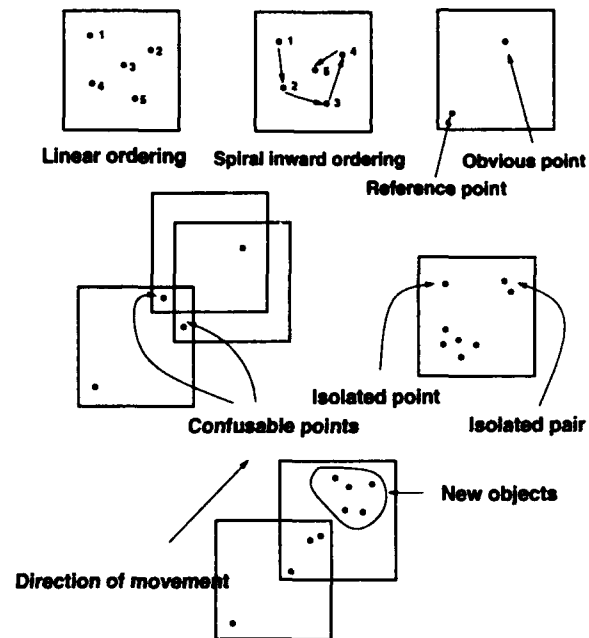


Figure 4: Examples of description language, only one of which (isolated point) this paper analyzes and implements

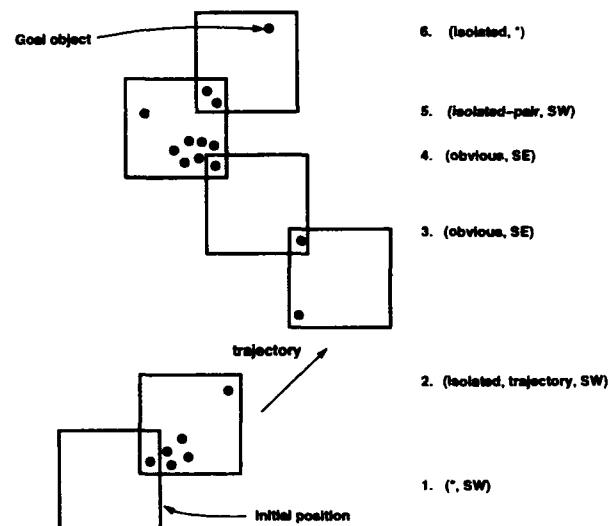


Figure 5: An example of navigation using a custom map, showing isolated point, obvious point, trajectory, and isolated pair

3.1 Calculating the isolated point

Our algorithm to compute the most isolated point in a scene consisting of n point-like objects uses the concept of *mutual neighborhoods*[Gowda and Krishna, 1978]. (Several other definitions of "isolated" led to unstable performance or costly computations.) The algorithm is as follows:

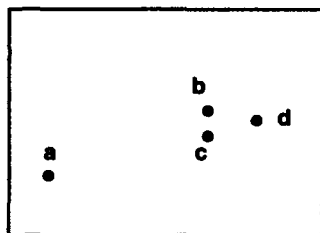


Figure 6: Finding the most isolated point in a window according to the *mnv* algorithm

	a	b	c	d
a	0	5	4	6
b	5	0	2	3
c	4	2	0	4
d	6	3	4	0

Table 2: The *mnv* matrix

1. For each pair of points in the scene compute the *mnv* (mutual neighborhood value). The *mnv* of two points *a* and *b* is the sum of two numbers, representing the order of how close *b* is to *a* and the order of how close *a* is to *b* relative to all the other objects. For example, if *b* is 2nd closest point to *a*, and *a* is 3rd closest point to *b*, then the *mnv* is 5. The result is stored in a $n \times n$ matrix where n is the number of visible objects. The objects in figure 6 have the *mnv* matrix of table 2.
2. For each column of the *mnv* matrix, find the smallest value greater than 0. This value is the *mnv* value between this particular object (that correspond to this particular column) and its "closest" neighbor. Call this value the "c-value" of this particular object. So in our example, the c-values for *a*, *b*, *c*, and *d* are 4, 2, 2, and 3 respectively.
3. The object (column) that has largest c-value is the most isolated point in the scene. In our example, *a* has the largest c-value(4) and therefore it is the most isolated point. Note that *c-value* is a small integer of value at most n .

Using the isolated point concept as the "connected" definition, we can form an "isolated" parkway of paths between isolated landmarks.

3.2 Trajectory traversal

The basic idea in trajectory method is to "slide" the view window along the direction formed by

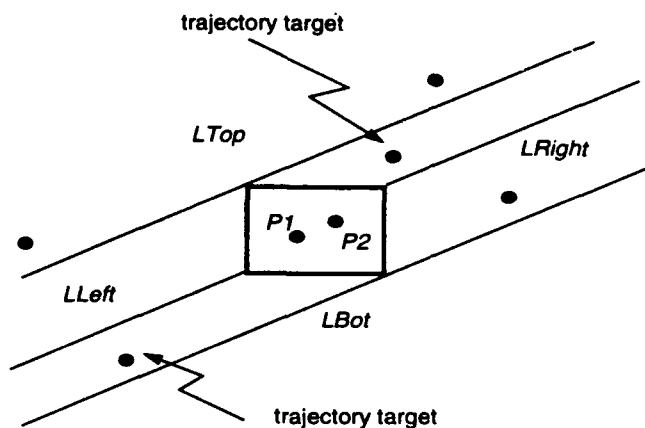


Figure 7: Formulation of Trajectory

two objects within the view window and to see which object gets encountered by the window first. Since the method we are using to identify a landmark in a view window is by selecting the isolated landmark, the two objects we use are the reference point and the isolated point. The directions formed by these two points, the positive and the negative directions, leads to at most two trajectory goal points. Note that we are not establishing any absolute coordinates for the navigator's movement, but in fact the navigator's movement is based on local orientations formed by landmark pairs. The implementation of trajectories is done by geometrically subdividing the navigation plane based on the navigator's view window position and the direction of the trajectory movement, and then computing to see which of the outstanding object is closest. In figure 7, we see a view window centered around two objects, $P_1(x_1, y_1)$, and $P_2(x_2, y_2)$. The directions formed by these two points, P_1 and P_2 , is defined by the slope m of the line that passes through these two points. The trajectory goal point, is then the closest object to this window along the directions (left and right) defined by m , bounded by the two lines, *LTop* and *LBot*. Therefore, for each pair of landmarks, there can be up to two different trajectories. When the navigator is moving right, two classes of objects are considered. The first class is the objects that are enclosed by the open polygon bounded by *LTop*, *LRight*, and the "north" part of the view window. The second class is the objects that are enclosed by the open polygon bounded by *LRight*, *LBot*, and the "east" part of the view window. From the objects that are in these two classes, the one which is closest to the view win-

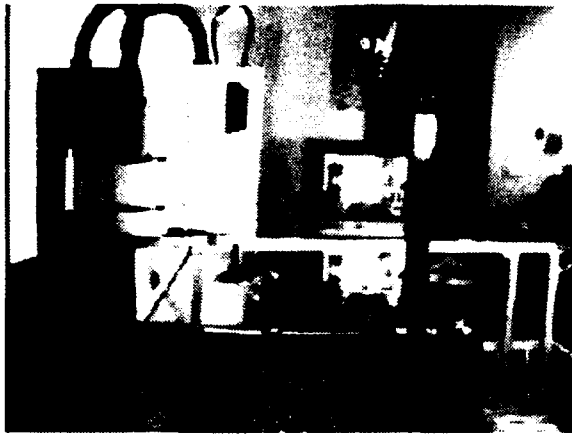


Figure 8: The map-maker and the navigator

dow frame is the trajectory goal object. When the navigator is moving left, two classes of objects are considered, similarly with respect to *LLeft*.

3.3 Implementation

We have implemented our map-maker and navigator using an IBM 7575 SCARA arm, two CCD cameras, PIPE, which is a high speed real-time image processor, and a SUN-4 workstation for high level control of the navigator and as the map-maker. Figure 8 shows the configuration of the map-maker and the navigator. The map-maker is comprised of one CCD camera located at a position that can capture the whole workspace of the navigator. This camera is attached to the PIPE which grabs the image and sends it to the SUN-4 workstation which runs the "map-making" program based on the centroid information of the scattered objects. The omniscience assumption of the map-maker requires that the image captured by the global camera correctly be used to generate the position of each object. To account for the image distortions due to translation, rotation, and perspective, we use a simple geometric calibration matrix A that transforms the homogeneous world coordinates $(X, Y, Z, 1)$ into the homogeneous camera coordinates $(U, V, 1)$ [Fu *et al.*, 1987].

Figure 11 shows the user interface program of the map-maker, running on the *X-Window* system. When the start and goal objects are chosen, the map-maker computes the optimal path in terms

of "the most isolated point" in each view window of the navigator. In the screen, the open dot at the lower right corner is the starting position and the open dot on the left side of the screen is the goal position. Each of the small rectangle represent the projected navigator's view along its path. For example, the rectangle in the lower right corner of the screen that contains the starting point is the initial view of the navigator. The line segment in each of these boxes connect the landmarks which are the most isolated points in the robot's path. Finally, the map-maker generates a file called "custommap" which contains the list of directional instructions for the navigator. The navigator is comprised of a second camera attached to the IBM robot arm. This camera is also connected to the PIPE for image processing of each scene as the navigator moves along. For each directional instruction in the custom map, the derivation of the most isolated point and the amount of movement of the robot for the corresponding instruction is computed by the SUN-4 workstation. It then sends out low level instructions to the IBM arm controller for the actual movement.

4 Error modeling

4.1 Reliability of isolated landmarks

The navigation using "custommap" was tested for various types of populations and start/goal positions. We discovered that the navigator tends to fail in subareas of the environment where the objects are highly populated. To explain this phenomenon, we did a statistical experiment, since the non linear definition of "isolated" defies analytic solution. We started with a randomly populated window with n objects. To simulate the inherent error of the navigator's position estimation of the visible objects, the position information, (x, y) , of each of the n objects was associated by a 2 dimensional Gaussian probability distribution. The output of such association produces a distorted position information, $(x + \epsilon_x, y + \epsilon_y)$. Then we applied the isolated point algorithm on the view window with the distorted positional information to see if the algorithm still identifies the correct landmark. The reliability of the isolated point in a view window with n objects is measured by the probability of achieving correct isolated point. This is approximated by the ratio $\frac{C}{N}$, where N is

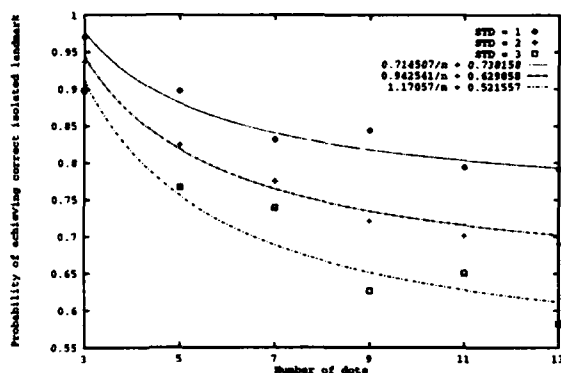


Figure 9: Reliability of isolated landmark decreases as the number of neighboring objects increases

the number of tries and C is the number of times when the algorithm identifies the correct isolated point. We ran this test with $N = 1000$, with $n = 3, 5, 7, 9, 11, 13$. Figure 9 shows the results. The data points near the top line indicates the reliability measures of the isolated point when the standard deviations, σ_x, σ_y , of the position estimations are equal to 1% of the window width W (of window size $W \times W$). The data points near middle and bottom line correspond to the reliability measures of isolated point when standard deviation are 2% and 3% of the window width, respectively. As we can see, the reliability of an isolated point decreases as the number of neighboring points increases. Note also that as the positional error increases (standard deviation of error function increases), the reliability decreases. After experimentation, we modeled the reliability of isolated point with the following equation, which states that the reliability is inversely proportional to the number of neighboring objects.

$$R = \frac{a\sigma + b}{n} + c\sigma + d \quad (1)$$

This equation was fitted with the data points using *Mathematica* package[Wolfram, 1988]. The result of the curve fitting is also shown in figure 9.

Figure 10 is a visualization of this reliability measures in original world of figure 1. Each "isolated point" candidate has a box around it. The size of the each box is defined as $W \times R$, where, $W \times W$

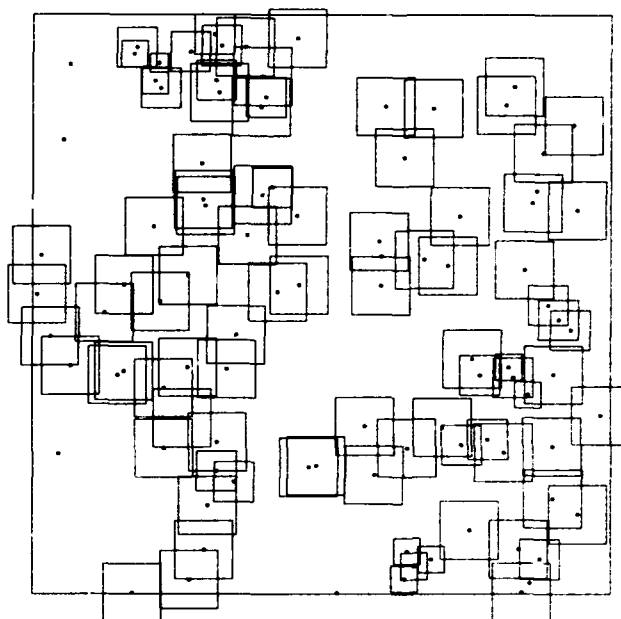


Figure 10: Reliability of each isolated landmark, illustrated graphically; larger boxes indicate higher reliability

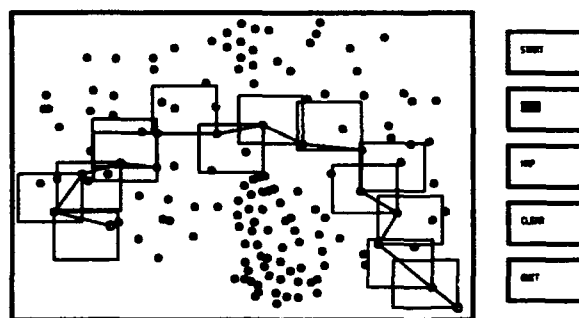


Figure 11: The user interface program for the map-maker, in which the world of dots has been captured by the camera and displayed. The command buttons are shown on the right. Also shown is the computed optimal path from a user-selected source to a user-selected goal.

is the view window size of the navigator, and R is the reliability of the isolated landmark. Basically, the larger the box is, the more reliable the object is as an isolated landmark. Therefore, during the derivation of the optimal path, objects with larger reliability windows are favored. Figure 11 shows the generated path that favors "reliable" isolated landmarks. Notice the navigator's path has detoured to avoid the highly cluttered area.

4.2 Reliability of trajectories

The reliability of the trajectory method depends on whether or not the trajectory goal is accurately achieved by the navigator's movement. Note that the error involved in achieving a wrong object during a trajectory movement is strictly one dimensional, as opposed to the two dimensional area involved in the error of "the most isolated-point" method. In the isolated point method, the neighboring objects surrounding the actual isolated point contribute to the error. However, the trajectory landmark needs to be distinct in only one direction - the direction of the navigator's trajectory movement.

The errors in trajectory traversal can come from sensor error in determining positional information of objects, or from translation or rotational error of the view window during navigation. The first of these three sources produce a static error, meaning that the error does not change with the traveling distance of the navigator. But the errors due to the second and the third sources are dynamic, meaning that the error grows as the trajectory distance increases.

First, let us examine the static error. Let X_i and X_j denote the distance from the view window frame to objects P_i and P_j , respectively. Assume that X_i and X_j follow some probability distributions with density functions $f(x_i)$ and $f(x_j)$ respectively. Since X_i and X_j are independent, the joint density function is given by,

$$f(x_i, x_j) = f(x_i)f(x_j)$$

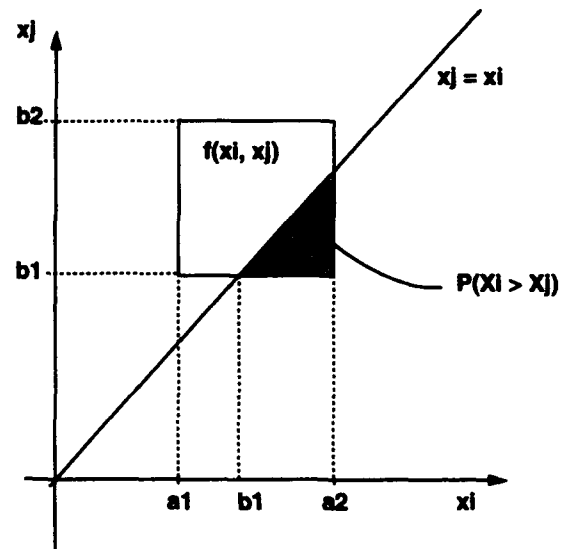
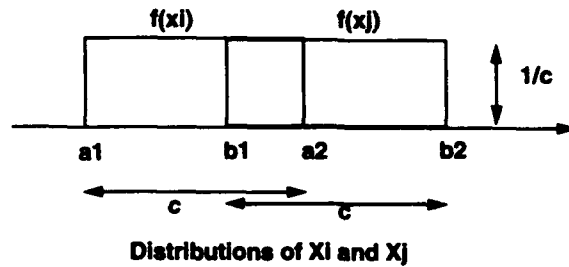
Then, the probability that object P_j will be reached before object P_i is given by:

$$P(X_i > X_j) = \iint_{x_i > x_j} f(x_i, x_j) dx_i dx_j$$

For simplicity in this paper, let us assume $f(x_i)$ and $f(x_j)$ are uniform distribution density function (other error models have also been analyzed) as in figure 12.

$$X_i \sim U(a_1, a_2) \text{ where } a_2 - a_1 = c$$

$$X_j \sim U(b_1, b_2) \text{ where } b_2 - b_1 = c$$



Joint distribution of X_i and X_j

Figure 12: Uniform joint probability distribution

Then the probability that B will be reached before A is given by $P(X_i > X_j)$, which is the area of the box under the line $x_j = x_i$ in figure 12.

$$\begin{aligned} P(X_i > X_j) &= \iint_{x_i > x_j} f(x_i)f(x_j) dx_i dx_j \\ &= \frac{1}{2c^2}(a_2 - b_1)^2 \end{aligned} \quad (2)$$

As stated earlier, errors due to the second and the third sources are dynamic, meaning that the error will grow as the trajectory distance increases. To model this, we use a variable size σ that is a function of distance. For example, if object P_j is 10 times farther away from the window than object P_i , then the distribution of P_j should be 10 times more dispersed than that of P_i . The question is what the "initial" value of the σ is. Let us assume that during the trajectory movement, as the navigator travels a distance equal to the window size (W), the dynamic error (in the case of Gaussian distribution, σ), is bounded by the size of the blob (radius r). This

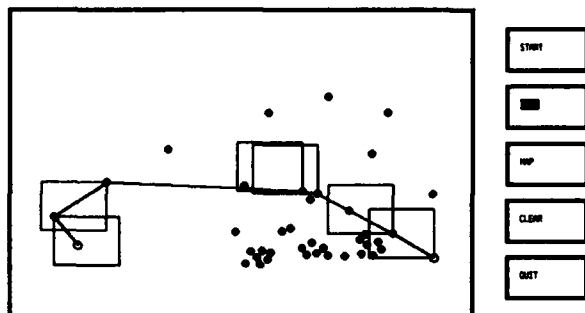


Figure 13: Combined path of Parkway traversal and Trajectory traversal

means that when the navigator travels a distance equal to its window size, the positional error can be as big as the blob size. Let the ratio between the blob radius and the window size be $1/B$ and the distance traveled be d . Then,

$$\sigma = \frac{dr}{W} = \frac{d}{B}$$

Combined navigation using parkways and trajectories both weighted by reliability is shown in figure 13. The corresponding custom map for the generated path is : ((isolated, SE), (isolated, trajectory, SE), (isolated, SE), (isolated, trajectory, NE), (isolated, NW), (isolated, *)).

5 Discussion

5.1 Tie breaking heuristics

Sometimes, particularly in sparse areas, there can be more than one winner in the isolated landmark method. In this case, two or more landmarks in the navigator's view window will seem equally isolated to the navigator, i.e., they have the same *mnv c-values* described previously. To break the tie, we use a heuristic to choose the isolated landmark that is the farthest away from the current reference point. This method is based on the observation that the navigator tends to travel towards the goal (and away from the starting location) at any instance of navigation. Therefore, by selecting the farthest landmark from the current reference point, the navigator usually moves closer to the goal. This is in contrast to a more usual search heuristic, which would select the landmark closest to the goal. The reason we use the former is that the global position of the goal (or even of the two competing landmarks) is unknown to the topologically driven navigator.

5.2 Definition of optimal path

Currently, we have cost functions that estimate the distance of navigation path D and the unreliability of navigation path R . One of map-maker's responsibility is to generate a path that either minimizes D , or maximizes R . Unfortunately, in some environments, these two cost estimates are in a direct conflict of each other. For example, if the shortest distance path involves in a highly cluttered area, the reliability would be low. Conversely, a reliable path that avoids highly cluttered areas may force the navigator to detour around a shortest path. Here, we suggest a third function C that compromises D and R , defined as: $C = \log D - \log R$. Using C as our cost estimate for travel, we can apply Dijkstra's shortest path algorithm to derive a path that minimizes the D/R ratio. The generated path will tend to favor short path and sparsely populated areas. Note that R is a function of not only n (population) but also σ (position estimate error) as in equations 1 and 2. If σ is small, the path will resemble the D -path and if σ is large, the path will resemble the R -path. This means that if the navigator's metric ability within its window is good, the optimal path will be the metrically shortest path. On the other hand, if the navigator's metric ability is poor, the optimal path will be the one that least confuses the navigator. In figure 13, note that the generated path neither passes through a highly cluttered areas nor it detours too much from the metrically shortest path.

5.3 Context-based landmarks

The shortest path in a parkway network does not guarantee the shortest travel path for the navigator because the cost (distance traveled) of achieving a landmark depends on the current "context" of the landmark. For example, consider a path generated within a parkway. Each arc of the path represents the distance between a reference point and a subsequent landmark. However, depending on the which corner of the view window (SE, SW, NE, NW) the new landmark is placed, the actual traveled distance of the navigator may or may not be equal to the arc length. In an extreme situation, we can visualize a parkway path that zigzags but the actual traveling movement of the navigator is linear and the traveled distance is much smaller (see

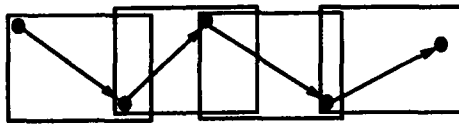


Figure 14: Zigzag arrows indicate the physical distances between landmarks, whereas the actual movement of the navigator is almost linear

figure 14.) This is analogous to a driver following a series of landmarks, such as buildings, in which the physical distances between landmarks is not equal to the odometer reading. Using our navigator model, each landmark has 4 different contexts, corresponding to the SE, SW, NE, and NW corners of the navigator's view window. In order to implement context-based landmark following, data structures to represent parkways, trajectories and the cost matrix, must be modified. Instead of n nodes in the parkway network, we now have $4n$ nodes, namely, the 4 ways each landmark can be "seen" by the navigator. The size of the cost matrix grows by a factor of 4^2 , but stays very sparse. Out of $16n^2$ cells, at most $12n$ are used. Therefore, a sparse matrix representation is needed for storage efficiency. Time complexity of the search algorithm is also increases, but by a factor of 4^3 since Dijkstra takes $O(n^3)$.

6 Future work

Future work will include a more elaborate description language that can fully represent the navigational scenes; statistical analysis to help decide on which "vocabulary" of the description language to use for each situation; error recovery schemes for the navigator to avoid, detect, and to correct for errorful situations; refinement of reliability of the directional instructions, and an increase in the navigator's degrees of freedom for a more general modeling of the real world navigation.

References

- [Abella, 1991] Alica Abella. Extracting geometric shapes from a set of points. In *Proc. Image Understanding Workshop*, 1991.
- [Fu et al., 1987] K.S. Fu, R.C. Gonzalez, and Lee C.S.G. *Robotics: Control, Sensing, Vision, and Intelligence*. McGraw Hill, 1987.
- [Gowda and Krishna, 1978] K. Chidananda Gowda and G. Krishna. Agglomerative clustering using the concept of mutual nearest neighborhood. *Pattern Recognition*, 1978.
- [Kender and Leff, 1989] John R. Kender and Avram Leff. Why direction-giving is hard: The complexity of linear navigation by landmarks. *IEEE Transactions on Systems, Man, and Cybernetics*, 1989.
- [Kender et al., 1990] John R. Kender, Il-Pyung Park, and David Yang. A formalization and implementation of topological visual navigation in two dimensions. In *SPIE International Symposia*, 1990.
- [Kuipers, 1978] B. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 1978.
- [Park and Kender, 1992] Il-Pyung Park and John R. Kender. Qualitative navigation using isolated landmarks. In *SPIE International Symposia*, 1992.
- [Park, 1991] Il-Pyung Park. Towards automatic vehicle navigation. Technical report, Columbia University, Department of Computer Science, Technical Report, 1991.
- [Park, 1993] Il-Pyung Park. *Qualitative Environmental Navigation: Theory and Practice*. PhD thesis, Columbia University, (Forthcoming), 1993.
- [Streeter et al., 1985] L. A. Streeter, D. Vitello, and S. A. Wonsiewica. How to tell people wherer to go: Comparing navigational aids. *International Journal of Man-Machine Studies*, 1985.
- [Wolfram, 1988] Stephen Wolfram. *Mathematica*. Addison-Wesley, 1988.

Section X

Visual Learning

Visual Learning of Object Models from Appearance

Hiroshi Murase and Shree K. Nayar

Department of Computer Science
Columbia University
New York, N.Y. 10027 *

Abstract

We address the problem of automatically learning object models for recognition and pose estimation. In contrast to the traditional approach, we formulate the recognition problem as one of matching appearance rather than shape. The appearance of an object in a two-dimensional image depends on its shape, reflectance properties, pose in the scene, and the illumination conditions. While shape and reflectance are intrinsic properties of an object and are constant, pose and illumination vary from scene to scene. We present a new compact representation of object appearance that is parametrized by pose and illumination. For each object of interest, a large set of images is obtained by automatically varying pose and illumination. This large image set is compressed to obtain a low-dimensional subspace, called the eigenspace, in which the object is represented as a hypersurface. Given an unknown input image, the recognition system projects the image onto the eigenspace. The object is recognized based on the hypersurface it lies on. The exact position of the projection on the hypersurface determines the object's pose in the image. We have conducted experiments using several objects with complex appearance characteristics. These results suggest the proposed appearance representation to be a valuable tool for a variety of machine vision applications.

1 Introduction

One of the primary goals of an intelligent vision system is to recognize objects in an image and compute their pose in the three-dimensional scene. Such a recognition system has wide applications ranging

from autonomous navigation to visual inspection. For a vision system to be able to recognize objects, it must have models of the objects stored in its memory. In the past, vision research has emphasized on the use of geometric (shape) models [Besl and Jain 85] [Chin and Dyer 86] for recognition. In the case of manufactured objects, these models are sometimes available and are referred to as computer aided design (CAD) models. Most objects of interest, however, do not come with CAD models. Typically, a vision programmer is forced to select an appropriate representation for object geometry, develop object models using this representation, and then manually input this information into the system. This procedure is cumbersome and impractical when dealing with large sets of objects, or objects with complicated geometric properties. It is clear that recognition systems of the future must be capable of acquiring object models without human assistance. In other words, recognition systems must be able to automatically *learn* the objects of interest.

Visual learning is clearly a well-developed and vital component of biological vision systems. If a human is handed an object and asked to visually memorize it, he or she would rotate the object and study its appearance from different directions. While little is known about the exact representations and techniques used by the human mind to learn objects, it is clear that the overall appearance of the object plays a critical role in its perception. In contrast to biological systems, machine vision systems today have little or no learning capabilities. Hence, visual learning is now emerging as an topic of research interest [Poggio and Girosi 90] [Ullman and Basri 91] [Ikeuchi and Suehiro 92]. The goal of this paper is to advance this important but relatively unexplored area of machine vision.

Here, we present a technique for automatically learning object models from images. The appearance

*This research was supported in part by DARPA Contract No. DACA 76-92-C-0007 and in part by the David and Lucile Packard Fellowship.

of an object is the combined effect of its shape, reflectance properties, pose in the scene, and the illumination conditions. Recognizing objects from brightness images is therefore more a problem of *appearance matching* rather than shape matching. This observation lies at the core of our work. While shape and reflectance are *intrinsic properties* of the object that do not vary, pose and illumination vary from scene to scene. We approach the visual learning problem as one of acquiring a compact model of the object's appearance under different illumination directions and object poses. The object is "shown" to the image sensor in several orientations and illumination directions. This can be accomplished using, for example, two robot manipulators; one to rotate the object while the other varies the illumination direction. The result is a very large set of object images. Since all images in the set are of the same object, any two consecutive images are correlated to large degree. The problem then is to compress this large image set into a low-dimensional representation of object appearance.

A well-known *image compression* or coding technique is based on the Karhunen-Loeve transform [Oja 83] [Fukunaga 90]. This method computes the eigenvectors of an image set. The eigenvectors form an orthogonal basis for the representation of individual images in the image set. Though a large number of eigenvectors may be required for very accurate reconstruction of an object image, only a few eigenvectors are generally sufficient to capture the significant appearance characteristics of an object. These eigenvectors constitute the dimensions of what we refer to as the *eigenspace* for the image set. From the perspective of machine vision, the eigenspace has a very attractive property. When it is composed of all the eigenvectors of an image set, it is optimal in a *correlation* sense: If any two images from the set are projected onto the eigenspace, the distance between the corresponding points in eigenspace is a measure of the similarity of the images in the l^2 norm. In machine vision, the Karhunen-Loeve method has been applied primarily to two problems; handwritten character recognition [Murase et al. 81] and human face recognition [Sirovich and Kirby 87], [Turk and Pentland 91]. These applications lie within the domain of pattern classification and do not address the problem of learning or using complete parametrized models of the objects of interest.

In this paper, we develop a continuous and compact representation of object appearance that is parametrized by the variables, namely, object pose and illumination. This new representation is referred to as the *parametric eigenspace*. First, an image set of

the object is obtained by varying pose and illumination in small increments. The image set is then normalized in brightness and scale to achieve invariance to image magnification and the intensity of illumination. The eigenspace for the image set is obtained by computing the most prominent eigenvectors of the image set. Next, all images in the object's image set (the learning samples) are projected onto the eigenspace to obtain a set of points. These points lie on a *hypersurface* that is parametrized by object pose and illumination. The hypersurface is computed from the discrete points using the cubic spline interpolation technique. It is important to note that this parametric representation of an object is obtained *without* prior knowledge of the object's shape and reflectance properties. It is generated using just a sample of the object.

Each object is represented as a parametric hypersurface in two different eigenspaces; the universal eigenspace and the object's own eigenspace. The *universal eigenspace* is computed by using the image sets of all objects of interest to the recognition system, and the *object eigenspace* is computed using only images of the object. We show that the universal eigenspace is best suited for discriminating between objects, whereas the object eigenspace is better tuned for pose estimation. Object recognition and pose estimation can be summarized as follows. Given an image consisting of an object of interest, we assume that the object is not occluded by other objects and can be segmented from the remaining scene. The segmented image region is normalized in scale and brightness, such that it has the same size and brightness range as the images used in the learning stage. This normalized image is first projected onto the universal eigenspace to identify the object. After the object is recognized, the image is projected onto the object eigenspace and the location of the projection on the object's parametrized hypersurface determines its pose in the scene.

We have conducted several experiments to demonstrate the power of the parametric eigenspace representation. The fundamental contributions of this paper can be summarized as follows. (a) The parametric eigenspace is presented as a new representation of object appearance. (b) Using this representation, object models are automatically learned from appearance by varying pose and illumination. (c) Both learning and recognition are accomplished without prior knowledge of the object's shape and reflectance.

2 Visual Learning of Objects

In this section, we discuss the learning of object models using the parametric eigenspace representation.

First, we discuss the acquisition of object image sets. The eigenspaces are computed using the image sets and each object is represented as a parametric hypersurface. Throughout this section, we will use a sample object to describe the learning process. In the next section, we discuss the recognition and pose estimation of objects using the parametric eigenspace representation.

2.1 Normalized Image Sets

While constructing image sets we need to ensure that all images of the object are of the same size. Each digitized image is first segmented (using a threshold) into an object region and a background region. The background is assigned a zero brightness value and the object region is re-sampled such that the larger of its two dimensions fits the image size we have selected for the image set representation. We now have a scale normalized image. This image is written as a vector \hat{x} by reading pixel brightness values from the image in a raster scan manner:

$$\hat{x} = [\hat{x}_1, \hat{x}_2, \dots, \hat{x}_N]^T \quad (1)$$

The appearance of an object depends on its shape and reflectance properties. These are intrinsic properties of the object that do not vary. The appearance of the object also depends on the pose of the object and the illumination conditions. Unlike the intrinsic properties, object pose and illumination are expected to vary from scene to scene. If the illumination conditions of the environment are constant, the appearance of the object is affected only by its pose. Here, we assume that the object is illuminated by the ambient lighting of the environment as well as one additional distant light source whose direction may vary. Hence, all possible appearances of the object can be captured by varying object pose and the light source direction with respect to the viewing direction of the sensor. We will denote each image as $\hat{x}_{r,l}^{(p)}$ where r is the rotation or pose parameter, l represents the illumination direction, and p is the object number. The complete image set obtained for an object is referred to as the **object image set** and can be expressed as:

$$\{ \hat{x}_{1,1}^{(p)}, \dots, \hat{x}_{R,1}^{(p)}, \hat{x}_{1,2}^{(p)}, \dots, \hat{x}_{R,L}^{(p)} \} \quad (2)$$

Here, R and L are the total number of discrete poses and illumination directions, respectively, used to obtain the image set. If a total of P objects are to be learned by the recognition system, we can define the **universal image set** as the union of all the object image sets:

$$\begin{aligned} & \{ \hat{x}_{1,1}^{(1)}, \dots, \hat{x}_{R,1}^{(1)}, \hat{x}_{1,2}^{(1)}, \dots, \hat{x}_{R,L}^{(1)}, \\ & \hat{x}_{1,1}^{(2)}, \dots, \hat{x}_{R,1}^{(2)}, \hat{x}_{1,2}^{(2)}, \dots, \hat{x}_{R,L}^{(2)}, \\ & \dots \\ & \hat{x}_{1,1}^{(P)}, \dots, \hat{x}_{R,1}^{(P)}, \hat{x}_{1,2}^{(P)}, \dots, \hat{x}_{R,L}^{(P)} \} \end{aligned} \quad (3)$$

We assume that the imaging sensor used for learning and recognizing objects has a linear response, i.e. image brightness is proportional to scene radiance. We would like our recognition system to be unaffected by variations in the intensity of illumination or the aperture of the imaging system. This can be achieved by normalizing each of the images in the object and universal sets such that its average brightness is zero and the brightness variance is unity. This brightness normalization transforms each measured image \hat{x} to a normalized image x :

$$x = [x_1, x_2, \dots, x_N]^T \quad (4)$$

where:

$$x_n = \frac{1}{B} (\hat{x}_n - A) \quad (5)$$

$$A = \frac{1}{N} \sum_{n=1}^N \hat{x}_n, \quad B = \sqrt{\sum_{n=1}^N (\hat{x}_n - A)^2}$$

The above described normalizations with respect to scale and brightness give us normalized object image sets and a normalized universal image set. In the following discussion, we will simply refer to these as the object and universal image sets.

The images sets can be obtained in several ways. If the geometrical model and reflectance properties of an object are known, its images for different pose and illumination directions can be synthesized using well-known rendering algorithms. In this paper, we do not assume that object geometry and reflectance are given. Instead, we assume that we have a sample of each object that can be used for learning. One approach then is to use two robot manipulators; one grasps the object and shows it to the sensor in different poses while the other has a light source mounted on it and is used to vary the illumination direction. In our experiments, we have used a turntable to rotate the object in a single plane (see Fig. 1). This gives us pose variations about a single axis. A robot manipulator is used to vary the illumination direction. If the recognition system is to be used in an environment where the illumination (due to one or several sources) is not expected to change, the image set can be obtained by varying just object pose.

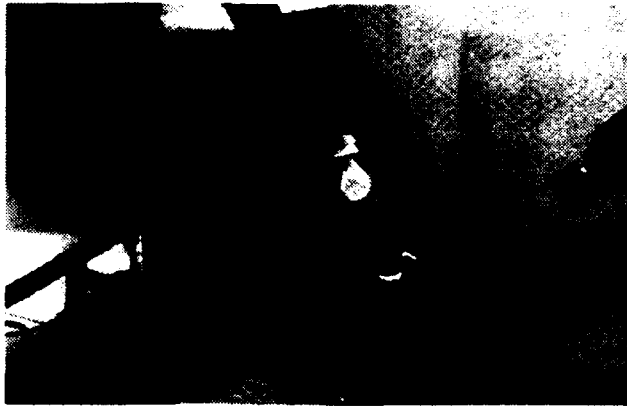


Figure 1: Setup used for automatic acquisition of object image sets. The object is placed on a motorized turntable.

2.2 Computing Eigenspaces

Consecutive images in an object image set tend to be correlated to a large degree since pose and illumination variations between consecutive images are small. Our first step is to take advantage of this correlation and compress large image sets into low-dimensional representations that capture the gross appearance characteristics of objects. A suitable compression technique is the Karhunen-Loeve expansion [Fukunaga 90] where the eigenvectors of the image set are computed and used as orthogonal basis functions for representing individual images. Though, in general, all the eigenvectors of an image set are required for the perfect reconstruction of an object image, only a few are sufficient for the representation of objects for recognition purposes. We compute two types of eigenspaces; the universal eigenspace that is obtained from the universal image set, and object eigenspaces computed from individual object image sets.

To compute the universal eigenspace, we first subtract the average c of all images in the universal set from each image. This ensures that the eigenvector with the largest eigenvalue represents the dimension in eigenspace in which the variance of images is maximum in the correlation sense. In other words, it is the most important dimension of the eigenspace. A new image set is obtained by subtracting the average image c from each image in the universal set:

$$\mathbf{X} \triangleq \{ \mathbf{x}_{1,1}^{(1)} - c, \mathbf{x}_{R,1}^{(1)} - c, \dots, \mathbf{x}_{R,L}^{(P)} - c \} \quad (6)$$

The image matrix \mathbf{X} is $N \times M$, where $M = RLP$ is the total number of images in the universal set, and N is the number of pixels in each image. To compute

eigenvectors of the image set we define the *covariance matrix* as:

$$\mathbf{Q} \triangleq \mathbf{X} \mathbf{X}^T \quad (7)$$

The covariance matrix is $N \times N$, clearly a very large matrix since a large number of pixels constitute an image. The eigenvectors \mathbf{e}_i and the corresponding eigenvalues λ_i of \mathbf{Q} are to be determined by solving the well-known eigenvector decomposition problem:

$$\lambda_i \mathbf{e}_i = \mathbf{Q} \mathbf{e}_i \quad (8)$$

All N eigenvectors of the universal set together constitute a complete eigenspace. Any two images from the universal image set, when projected onto the eigenspace, give two discrete points. The distance between these points is a measure of the difference between the two images in the correlation sense. Since the universal eigenspace is computed using images of all objects, it is the ideal space for discriminating between images of different objects.

Determining the eigenvalues and eigenvectors of a large matrix such as \mathbf{Q} is a non-trivial problem. It is computationally very intensive and traditional techniques used for computing eigenvectors of small matrices are impractical. Since we are interested only in a small number (k) of eigenvectors, and not the complete set of N eigenvectors, efficient algorithms can be used. In our implementation, we have used the *spatial temporal adaptive* (STA) algorithm proposed by Murase and Lindenbaum [Murase and Lindenbaum 92]. This algorithm was recently demonstrated to be substantially more efficient than previous algorithms. Using the STA algorithm the k most prominent eigenvectors of the universal image set are computed. The result is a set of eigenvalues $\{ \lambda_i \mid i = 1, 2, \dots, k \}$ where $\{ \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k \}$, and a corresponding set of eigenvector $\{ \mathbf{e}_i \mid i = 1, 2, \dots, k \}$. Note that each eigenvector is of size N , i.e. the size of an image. These k eigenvectors constitute the universal eigenspace; it is an approximation to the complete eigenspace with N dimensions. We have found from our experiments that less than ten dimensions of the eigenspace are generally sufficient for the purposes of visual learning and recognition (i.e. $k \leq 10$). Later, we describe how objects in an unknown input image are recognized using the universal eigenspace.

Once an object has been recognized, we are interested in finding its pose in the image. The accuracy of pose estimation depends on the ability of the recognition system to discriminate between different images

of the same object. Hence, pose estimation is best done in an eigenspace that is tuned to the appearance of a single object. To this end, we compute an object eigenspace from each of the object image sets. The procedure for computing an object eigenspace is similar to that used for the universal eigenspace. In this case, the average $c^{(p)}$ of all images of object p is computed and subtracted from each of the object images. The resulting images are used to compute the covariance matrix $Q^{(p)}$. The eigenspace for the object p is obtained by solving the system:

$$\lambda_i^{(p)} e_i^{(p)} = Q^{(p)} e_i^{(p)} \quad (9)$$

Once again, we compute only a small number ($k \leq 10$) of the largest eigenvalues $\{\lambda_i^{(p)} \mid i = 1, 2, \dots, k\}$ where $\{\lambda_1^{(p)} \geq \lambda_2^{(p)} \geq \dots \geq \lambda_k^{(p)}\}$, and a corresponding set of eigenvector $\{e_i^{(p)} \mid i = 1, 2, \dots, k\}$. An object eigenspace is computed for each object of interest to the recognition system.

2.3 Parametric Eigenspace Representation

We now represent each object as a hypersurface in the universal eigenspace as well as its own eigenspace. This new representation of appearance lies at the core of our approach to visual learning and recognition. Each appearance hypersurface is parametrized by two parameters; object rotation and illumination direction.

A parametric hypersurface for the object p is constructed in the universal eigenspace as follows. Each image $x_{r,l}^{(p)}$ (learning sample) in the object image set is projected onto the eigenspace by first subtracting the average image c from it and finding the dot products of the result with each of the eigenvectors (dimensions) of the universal eigenspace. The result is a point $g_{r,l}^{(p)}$ in the eigenspace:

$$g_{r,l}^{(p)} = [e_1, e_2, \dots, e_k]^T (x_{r,l}^{(p)} - c) \quad (10)$$

Once again the subscript r represents the rotation parameter and l is the illumination direction. By projecting all the learning samples in this manner, we obtain a set of discrete points in the universal eigenspace. Since consecutive object images are strongly correlated, their projections in eigenspace are close to one another. Hence, the discrete points obtained by projecting all the learning samples can be assumed to lie on a k -dimensional hypersurface that represents all possible poses of the object under all possible illumination directions. We interpolate the discrete points to obtain this hypersurface. In our implementation,

we have used a standard cubic spline interpolation algorithm [Press et al. 88]. Since cubic splines are well-known we will not describe them here. The resulting hypersurface can be expressed as:

$$g^{(p)}(\theta_1, \theta_2) \quad (11)$$

where θ_1 and θ_2 are the *continuous* rotation and illumination parameters. The above hypersurface is a compact representation of the object's appearance.

In a similar manner, a hypersurface is also constructed in the object's eigenspace by projecting the learning samples onto this space:

$$f_{r,l}^{(p)} = [e_1^{(p)}, e_2^{(p)}, \dots, e_k^{(p)}]^T (x_{r,l}^{(p)} - c^{(p)}) \quad (12)$$

where, $c^{(p)}$ is the average of all images in the object image set. Using cubic splines, the discrete points $f_{r,l}^{(p)}$ are interpolated to obtain the hypersurface:

$$f^{(p)}(\theta_1, \theta_2) \quad (13)$$

Once again, θ_1 and θ_2 are the rotation and illumination parameters, respectively. This continuous parameterization enables us to find poses of the object that are not included in the learning samples. It also enables us to compute accurate pose estimates under illumination directions that lie in between the discrete illumination directions used in the learning stage. Fig.2 shows the parametrized eigenspace representation of the object shown in Fig.1. The figure shows only three of the most significant dimensions of the eigenspace since it is difficult to display and visualize higher dimensional spaces. The object representation in this case is a curve, rather than a surface, since the object image set was obtained using a single illumination direction while the object was rotated about a single axis. The discrete points on the curve correspond to projections of the learning samples in the object image set. The continuous curve passing through the points is parametrized by the rotation parameter θ_1 and is obtained using the cubic spline algorithm.

3 Recognition and Pose Estimation

Consider an image of a scene that includes one or more of the objects we have learned using the parametric eigenspace representation. We assume that the objects are not occluded by other objects in the scene when viewed from the sensor direction, and that the image regions corresponding to objects have been segmented away from the scene image. First, each segmented image region is normalized with respect to

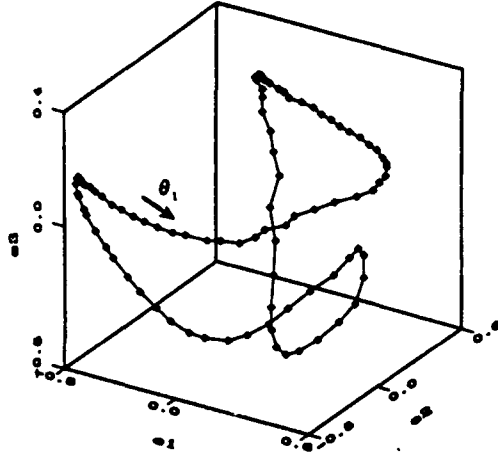


Figure 2: Parametric eigenspace representation of the object shown in Fig.1. Only the three most prominent dimensions of the eigenspace are displayed here. The points shown correspond to projections of the learning samples. Here, illumination is constant and therefore we obtain a curve with a single parameter (rotation) rather than a surface.

scale and brightness as described in section 2.1. This ensures that (a) the input image has the same dimensions as the eigenvectors (dimensions) of the parametric eigenspace, (b) the recognition system is invariant to object magnification, and (c) the recognition system is invariant to fluctuations in the intensity of illumination.

As stated earlier in the paper, the universal eigenspace is best tuned to discriminate between different objects. Hence, we first project the normalized input image y to the universal eigenspace. First, the average c of the universal image set is subtracted from y and the dot product of the resulting vector is computed with each of the eigenvectors that constitute the universal space. The k coefficients obtained are the coordinates of a point z in the eigenspace:

$$z = [e_1, e_2, \dots, e_k]^T (y - c) = [z_1, z_2, \dots, z_k]^T \quad (14)$$

The recognition problem then is to find the object p whose hypersurface the point z lies on. Due to factors such as image noise, aberrations in the imaging system, and digitization effects, z may not lie exactly on an object hypersurface. Hence, we find the object p that gives the minimum distance $d_1^{(p)}$ between its hypersurface $g^{(p)}(\theta_1, \theta_2)$ and the point z :

$$d_1^{(p)} = \min_{\theta_1, \theta_2} \| z - g^{(p)}(\theta_1, \theta_2) \| \quad (15)$$

If $d_1^{(p)}$ is within some pre-determined threshold value, we conclude that the input image is of the object p . If not, we assume that input image is not of any of the objects used in the learning stage. It is important to note that the hypersurface representation of objects in eigenspace results in more reliable recognition than if the object is represented as just a cluster of the points $g_{r,l}^{(p)}$ in eigenspace. The hypersurfaces of different objects can intersect each other or even be intertwined, in which cases, using nearest cluster algorithms could easily lead to incorrect recognition results.

Once the object in the input image y is recognized, we project y to the eigenspace of the object. This eigenspace is tuned to variations in the appearance of a single object and hence is ideal for pose estimation. Mapping the input image to the object eigenspace gives the k -dimensional point:

$$\begin{aligned} z^{(p)} &= [e_1^{(p)}, e_2^{(p)}, \dots, e_k^{(p)}]^T (y - c^{(p)}) \quad (16) \\ &= [z_1^{(p)}, z_2^{(p)}, \dots, z_k^{(p)}]^T \end{aligned}$$

The pose estimation problem may be stated as follows: Find the rotation parameter θ_1 and the illumination parameter θ_2 that minimize the distance $d_2^{(p)}$ between the point $z^{(p)}$ and the hypersurface $f^{(p)}$ of the object p :

$$d_2^{(p)} = \min_{\theta_1, \theta_2} \| z - f^{(p)}(\theta_1, \theta_2) \| \quad (17)$$

The θ_1 value obtained represents the pose of the object in the input image. Fig. 3(a) shows an input image of the object whose parametric eigenspace was shown in Fig. 2. This input image is not one of the images in the learning set used to compute the object eigenspace. In Fig. 3b, the input image is mapped to the object eigenspace and is seen to lie on the parametric curve of the object. The location of the point on the curve determines the object's pose in the image. Note that the recognition and pose estimation stages are computationally very efficient, each requiring only the projection of an input image onto a low-dimensional (generally less than 10) eigenspace. Customized hardware can therefore be used to achieve real-time (frame-rate) recognition and pose estimation.

4 Experimentation

We have conducted several experiments using complex objects to verify the effectiveness of the parametric eigenspace representation. This section summarizes some of our results. Fig. 1 in section 2 shows the set-up used to conduct the experiments reported here.

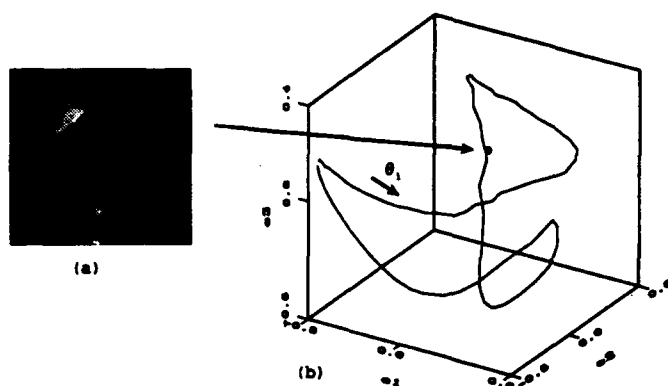


Figure 3: (a) An input image. (b) The input image is mapped to a point in the object eigenspace. The location of the point on the parametric curve determines the pose of the object in the input image.

The object is placed on a motorized turntable and its pose is varied about a single axis, namely, the axis of rotation of the turntable. The turntable position is controlled through software and can be varied with an accuracy of about 0.1 degrees. Most objects have a finite number of stable configurations when placed on a planar surface. For such objects, the turntable is adequate as it can be used to vary pose for each of the object's stable configurations.

We assume that the object is illuminated by the ambient lighting conditions of the environment that are not expected to change between the learning and recognition stages. This ambient illumination is of relatively low intensity. The main source of brightness is an additional light source whose direction can vary. In most of our experiments, the source direction was varied manually. We are currently using a 6 degree-of-freedom robot manipulator (see Fig. 1) with a light source mounted on its end-effector. This enables us to vary the illumination direction via software. Images of the object are sensed using a 512×480 pixel CCD camera and are digitized using an Analogics framegrabber board.

Table 1 summarizes the number of objects, light source directions, and poses used to acquire the image sets used in the experiments. For the learning stage, a total of 4 objects were used. These objects (cars) are shown in Fig. 4(a). For each object we have used 5 different light source directions, and 90 poses for each source direction. This gives us a total of 1800 images in the universal image set and 450 images in each object image set. Each of these images is automatically normalized in scale and brightness as described in sec-

tion 2. Each normalized image is 128×128 pixels in size. The universal and object image sets were used to compute the universal and object eigenspaces. The parametric eigenspace representations of the four objects in their own eigenspaces are shown in Fig. 4(b).

Table 1: Image sets obtained for the learning and recognition stages. The 1080 test images used for recognition are different from the 1800 images used for learning.

Learning Samples	Test Samples for Recognition
4 Objects	4 Objects
5 Light Source Directions	3 Light Source Directions
90 Poses	90 Poses
1800 Images	1080 Images

A large number of images were also obtained to test the recognition and pose estimation algorithms. All of these images are different from the ones used in the learning stage. A total of 1080 input (test) images were obtained. The illumination directions and object poses used to obtain the test images are different from the ones used to obtain the object image sets for learning. In fact, the test images correspond to poses and illumination directions that lie in between the ones used for learning. Each input image is first normalized in scale and brightness and then projected onto the universal eigenspace. The object in the image is identified by finding the hypersurface that is closest to the input point in the universal eigenspace. Unlike the learning process, recognition is computationally simple and can be accomplished on a Sun SPARC 2 workstation in less than 0.2 second.

To evaluate the recognition results, we define the *recognition rate* as the percentage of input images for which the object in the image is correctly recognized. Fig. 5(a) illustrates the sensitivity of the recognition rate to the number of dimensions of the universal eigenspace. Clearly, the discriminating power of the universal eigenspace is expected to increase with the number of dimensions. For the objects used, the recognition rate is poor if less than 4 dimensions are used but approaches unity as the number of dimensions approaches 10. In general, however, the number of dimensions needed for robust recognition is expected to increase with the number of objects learned by the system. It also depends on the appearance characteristics of the objects used. From our experience, 10 dimensions are sufficient for representing objects with

fairly complex appearance characteristics such as the ones shown in Fig. 4.

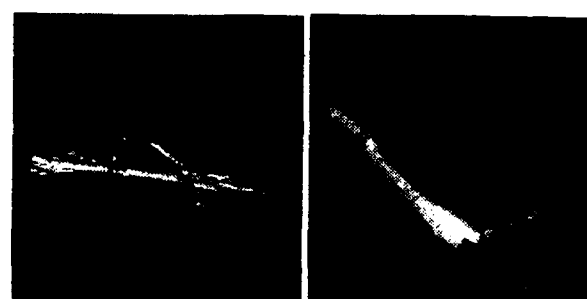
Finally, we present experimental results related to pose estimation. Once the object is recognized, the input image is projected onto the object's eigenspace and its pose is computed by finding the closest point on the parametric hypersurface. Once again we use all 1080 input images of the 4 objects. Since these images were obtained using the controlled turntable, the actual object pose in each image is known. Fig. 5(b, and (c) shows histograms of the errors (in degrees) in the poses computed for the 1080 images. The error histogram in Fig. 5(b) is for the case where 450 learning samples (90 poses and 5 source directions) were used to compute the object eigenspace. The eigenspace used has 8 dimensions. The histogram in Fig. 5(c) is for the case where 90 learning samples (18 poses and 5 source directions) were used. The pose estimation results in both cases were found to be remarkably accurate. In the first case, the average of the absolute pose error computed using all 1080 images is found to be 0.5 degrees, while in the second case the average error is found to be 1.2 degrees.

5 Conclusion

In this paper, we presented a new representation for machine vision called the parametric eigenspace. While representations previously used in computer vision are based on object geometry, the proposed representation describes object appearance. We presented a method for automatically learning an object's parametric eigenspace. Such learning techniques are fundamental to the advancement of visual perception. We developed efficient object recognition and pose estimation algorithms that are based on the parametric eigenspace representation. The learning and recognition algorithms were tested on objects with complex shape and reflectance properties. A statistical analysis of the errors in recognition and pose estimation demonstrate the proposed approach to be very robust to factors, such as, image noise and quantization. We believe that the results presented in this paper are applicable to a variety of vision problems. This is the topic of our current investigation.

References

- [Besl and Jain 85] P. J. Besl and R. C. Jain, "Three-Dimensional Object Recognition," *ACM Computing Surveys*, Vol. 17, No. 1, pp. 75-145, 1985.
- [Chin and Dyer 86] R. T. Chin and C. R. Dyer, "Model-Based Recognition in Robot Vision," *ACM Computing Surveys*, Vol. 18, No. 1, pp. March 1986.
- [Fukunaga 90] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, London, 1990.
- [Ikeuchi and Suehiro 92] K. Ikeuchi and T. Suehiro, "Recognizing Assembly Tasks using Face-Contact Relations," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 154-160, June 1992.
- [Murase et al. 81] H. Murase, F. Kimura, M. Yoshimura, and Y. Miyake, "An Improvement of the Auto-Correlation Matrix in Pattern Matching Method and Its Application to Handprinted 'HIRAGANA'," *Trans. IECE*, Vol. J64-D, No. 3, 1981.
- [Murase and Lindenbaum 92] H. Murase and M. Lindenbaum, "Spatial Temporal Adaptive Method for Partial Eigenstructure Decomposition of Large Images," *NTT Technical Report No. 6527*, March 1992.
- [Oja 83] E. Oja, *Subspace methods of Pattern Recognition*, Research Studies Press, Hertfordshire, 1983.
- [Poggio and Girosi 90] T. Poggio and F. Girosi, "Networks for Approximation and Learning," *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1481-1497, September 1990.
- [Press et al. 88] W. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C*, Cambridge University Press, Cambridge, 1988.
- [Sirovich and Kirby 87] L. Sirovich and M. Kirby, "Low dimensional procedure for the characterization of human faces," *Set of Images*, *Journal of Optical Society of America*, Vol. 4, No. 3, pp. 519-524, 1987.
- [Turk and Pentland 91] M. A. Turk and A. P. Pentland, "Face Recognition Using Eigenfaces," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586-591, June 1991.
- [Ullman and Basri 91] S. Ullman and R. Basri, "Recognition by Linear Combination of Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 10, pp. 992-1006, October 1991.



A

B

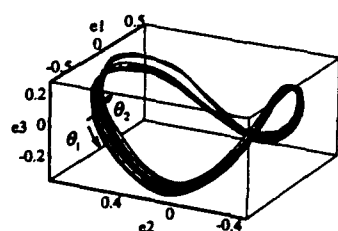


C

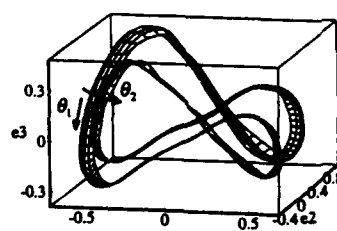


D

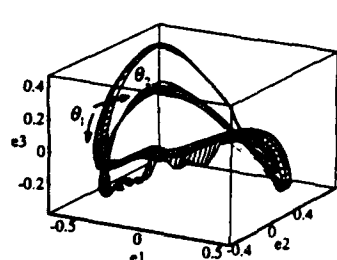
(a)



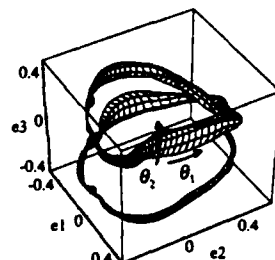
A



B



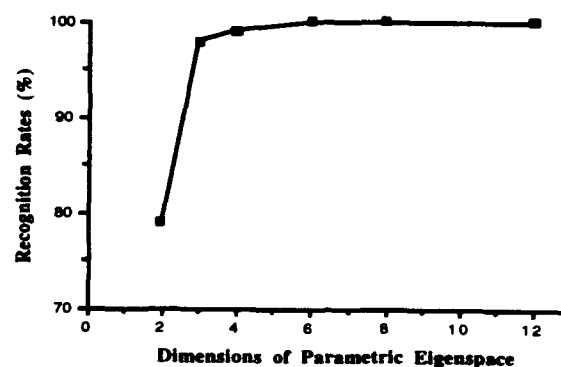
C



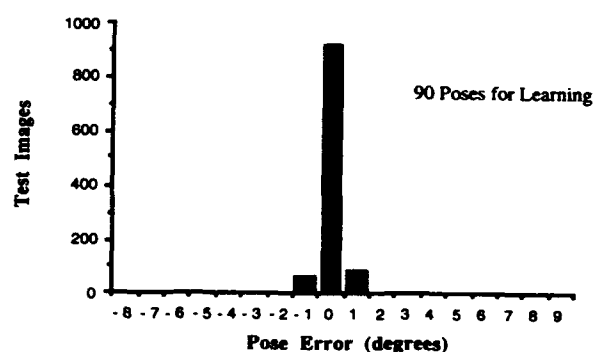
D

(b)

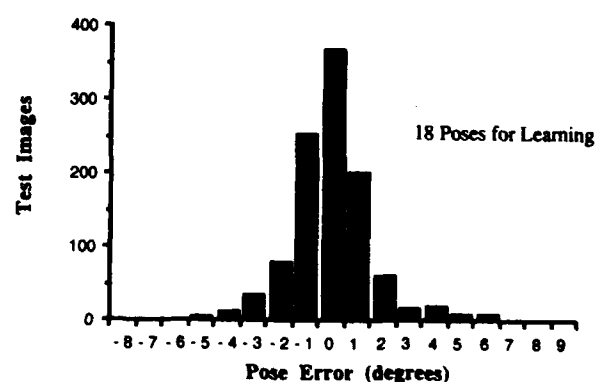
Figure 4: (a) The four objects used in the experiments. (b) The parametric hypersurfaces in object eigenspace computed for the four objects shown in (a). For display, only the three most important dimensions of each eigenspace are shown. The hypersurfaces are reduced to surfaces in three-dimensional space.



(a)



(b)



(c)

Figure 5: (a) Recognition rate plotted as a function of the number of universal eigenspace dimensions used to represent the parametric hypersurfaces. The recognition rates were computed using all 1080 input images detailed in Table 1. (b) Histogram of the error (in degrees) in computed object pose for the case where 90 poses are used in the learning stage. (c) Pose error histogram for the case where 18 poses are used in the learning stage. The average of the absolute error in pose for the complete set of 1080 test images is 0.5 in the first case and 1.2 in the second case.

Statistical Properties of Learning Recognition Strategies

Bruce A. Draper

Dept. of Computer Science
University of Massachusetts
Amherst, MA., USA. 01003*

Abstract

The Schema Learning System (SLS) automates the construction of knowledge-directed recognition strategies by learning object-specific strategies that integrate many different visual procedures. Starting from training images with ground truth object positions, an object model, and a library of visual procedures, SLS learns which procedures aid in the recognition of a particular object, and which are unreliable, unnecessary or too expensive. It then builds a strategy for controlling selected procedures that will reliably recognize the target at a minimum cost.

This paper does not present the algorithm, published previously, by which SLS infers recognition strategies. Instead, it focuses on SLS's role as an integrator of disparate visual procedures, and on an analysis, based on a lemma by Valiant [1984], giving a probabilistic upper bound on the likelihood that a strategy will fail to recognize an object in a test image.

1 Introduction

Most knowledge-directed vision systems are hand-crafted to recognize a fixed set of objects within a known context. Generally, the programmer or knowledge engineer who constructs them begins with an intuitive notion of how each object might be recognized, a notion which is refined by trial-and-error. Eventually the programmer finds a combination of features (e.g. color, shape or context) and methods (e.g. geometric model matching, minimum-distance classification or generalized Hough transforms) that allow the objects to be reliably identified within the domain. In this way, separate, hand-crafted strategies are constructed for all the objects in a domain.

Unfortunately, there is a growing consensus that human engineering is not cost-effective for many applications. Moreover, even when hand-crafted systems

can be constructed, there is no way to ensure their validity, since their performance, in terms of accuracy and reliability, is unknown, and there is no objective test for comparing one strategy to another. Worst of all, when the domain is changed, hand-crafted systems often have to be rebuilt from scratch.

If we look at the knowledge engineering process more closely, we find that it is convenient to draw a distinction between object models and the visual processes used to extract them. In a typical application, a knowledge engineer starts with a model of the object(s) to be recognized. In some cases, the model is a complete geometric description of the shape of an object, as in a wire-frame or generalized-cylinder model. In other cases, the model describes other features, such as the color or texture of an object. In still other cases, the model may be in the form of constraints.

Whatever the model, the knowledge engineer's task is to select visual procedures for matching the object model to image data or, equivalently, reconstructing the target object from the image data consistent with the constraints of the object model. Visual procedures are selected based on both the form of the object model and knowledge of the domain, such as lighting conditions and the likelihood of occlusion.

In general, recognition strategies require more than a single visual procedure. Most visual procedures are designed to solve particular subtasks, and several must be sequenced together in order to reconstruct an object model. For example, Figure 1 shows several alternative methods for finding the pose of a road sign, each of which depends on one or more intermediate representations of the data. To build a recognition strategy, a knowledge engineer must find a series of visual procedures that lead from the original image data to the desired representation. In addition, most visual procedures are prone to occasional failures, so system designers must consider how much redundancy to include in each strategy.

Despite the labor-intensive knowledge-engineering process, many knowledge-directed systems have been built and successfully demonstrated, e.g. [Draper, et.

*This work was supported by Rome Labs under contract F30602-91-C-0037 and by DARPA/TACOM under contract DAAE07-91-C-R035

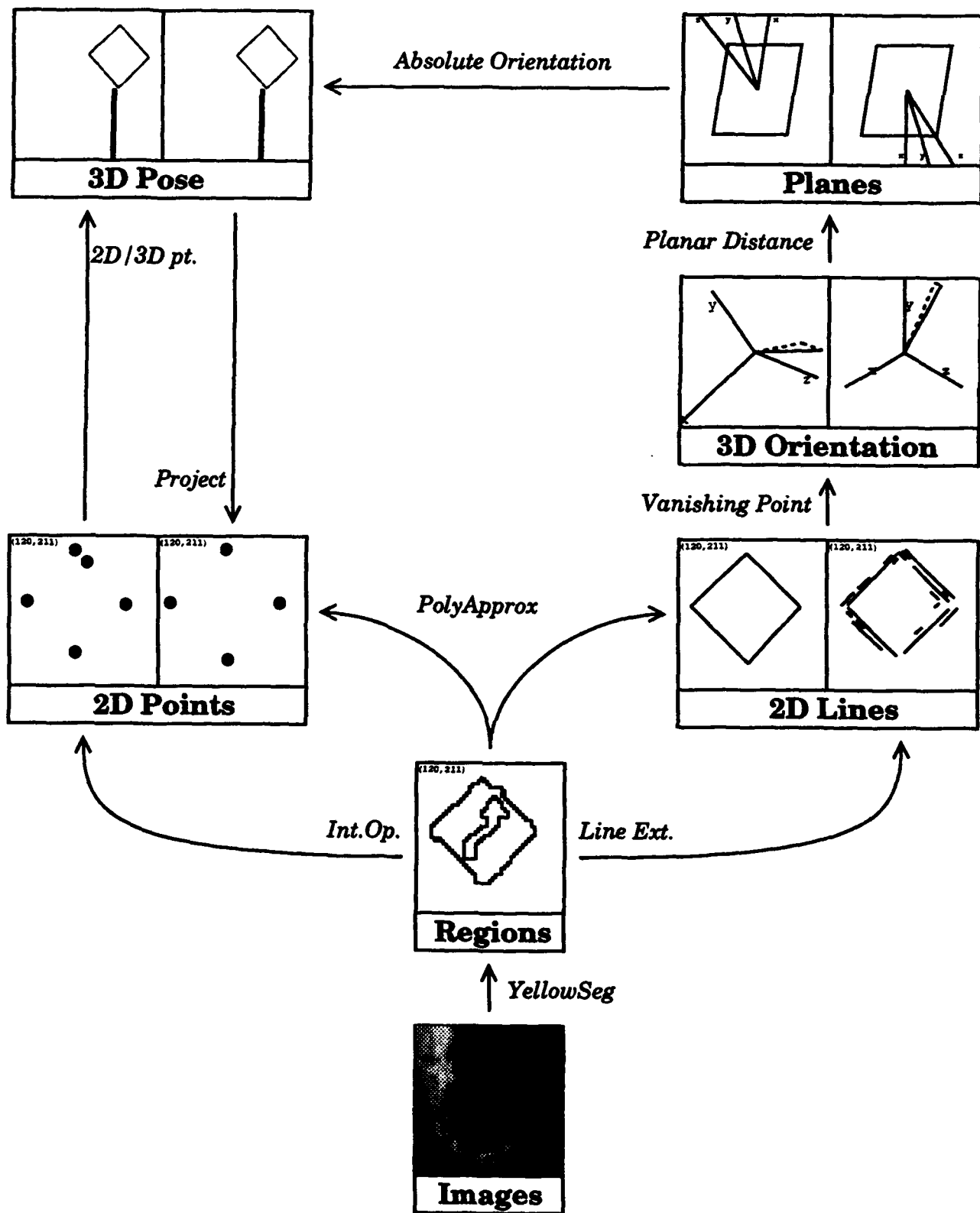


Figure 1: Several methods for determining the pose of a roadsign.

al., 1989, McKeown, et. al., 1985, Hwang, et. al., 1986]. In general, the success of these systems can be traced to a "small world" assumption, in which the number of objects in the domain are few, the constraints on their descriptions are tight, and a complete world model is at least a possibility. Consequently, knowledge engineers are able to select the appropriate visual procedures for the domain and object models, and to devise effective control procedures for applying them. However, as the scope of a system broadens towards a domain-independent, general-purpose system, system designers are faced with an unfortunate dilemma: either they craft more and more special-purpose strategies, an option that quickly becomes infeasible in terms of labor, or they generalize their strategies to recognize more and more objects under a wider range of circumstances. In the latter case, constraints on object descriptions become looser to account for wider variability, the system can make fewer assumptions about the types of image descriptions necessary for matching, and the complexity of matching increases substantially.

2 Knowledge Base Construction

In general, the difficulty and expense of knowledge base construction has relegated knowledge-directed vision to the laboratory, where the domain can be restricted to a few objects in a controlled context. Although artificial intelligence researchers handle this problem by extracting knowledge from experts, this scenario does not apply to computer vision. Vision researchers have therefore concentrated instead on knowledge base specification. The SPAM project developed a high-level language for describing objects [McKeown, et. al., 1989] and a series of tools designed to automate pieces of the knowledge acquisition process [Harvey, et. al., 1992]. The UMass Schema System divided both the interpretation process and the knowledge base by object, increasing modularity and making the system easier to modify [Draper, et. al., 1989]. Work in Japan has involved both automatic programming efforts and higher-level languages for specifying image operations [Matsuyama 1989].

3 The Schema Learning System (SLS)

The Schema Learning System (SLS) presents a different solution to the knowledge base construction problem. SLS is a system that automatically learns object-specific or task-specific recognition strategies from object models, training images and a library of visual procedures, as shown in Figure 2. The idea behind SLS is that a general-purpose vision system can be constructed of hundreds of special-purpose recognition strategies, each learned from experience, rather than from a single, highly-general, and therefore highly inefficient, recognition strategy.

As described in this paper, SLS's task is to learn strategies that recognize a particular object and recover its three-dimensional position and orientation. (SLS can also be used for two-dimensional recognition, or for learning predicate strategies that determine if an object is present or not.) It is trained on images in which the position and orientation of the target object are known. Its goal is to learn a control strategy for invoking visual procedures from its library that minimizes the expected cost of recognizing the target object across the training images. As will be discussed in the next section, it is also able to predict the expected cost of each recognition strategy and to bound the probability of failure.

3.1 The SLS Process Model

SLS is similar to a blackboard system in that recognition is viewed as a process of applying visual procedures (VPs) to hypotheses. Hypotheses are representations of the image or 3D world such as points, lines, regions or surfaces; visual procedures are algorithms from the image understanding literature such as line extraction or geometric model matching [Beveridge and Riseman, 1992]. Recognition strategies take the place of dynamic schedulers in traditional blackboard systems, selecting which procedure(s) to apply at each step.

Therefore, recognition can be described as a branching sequence of VP invocations. The sequence begins when data arrives, typically in the form of image hypotheses¹. Visual procedures are applied to images, producing low-level hypotheses such as points, lines or regions. New VPs are then applied to these low-level hypotheses, transforming them into more abstract hypotheses. Still more VPs are applied to these hypotheses in a repeating cycle, until eventually goal-level hypotheses are created.

3.1.1 Transformation Procedures (TPs)

Unlike most blackboard systems, however, SLS refines its processing model by dividing visual procedures into two classes, *transformation procedures* (TPs) and *feature measurement procedures* (FMPs)². Transformation procedures transform old hypotheses into new hypotheses at a higher level of representation. Examples include vanishing point analysis, which creates surface orientation hypotheses from pencils of image lines, and stereo line matching, which creates world (3D) line hypotheses from pairs of image (2D) line hypotheses. Feature measurement procedures, by way of comparison, measure properties of hypotheses without changing their underlying representations.

¹Typically, but not necessarily. Active strategies may invoke procedures to acquire image data.

²We will still use the generic term *visual procedures* (VPs) when referring to both TPs and FMPs.

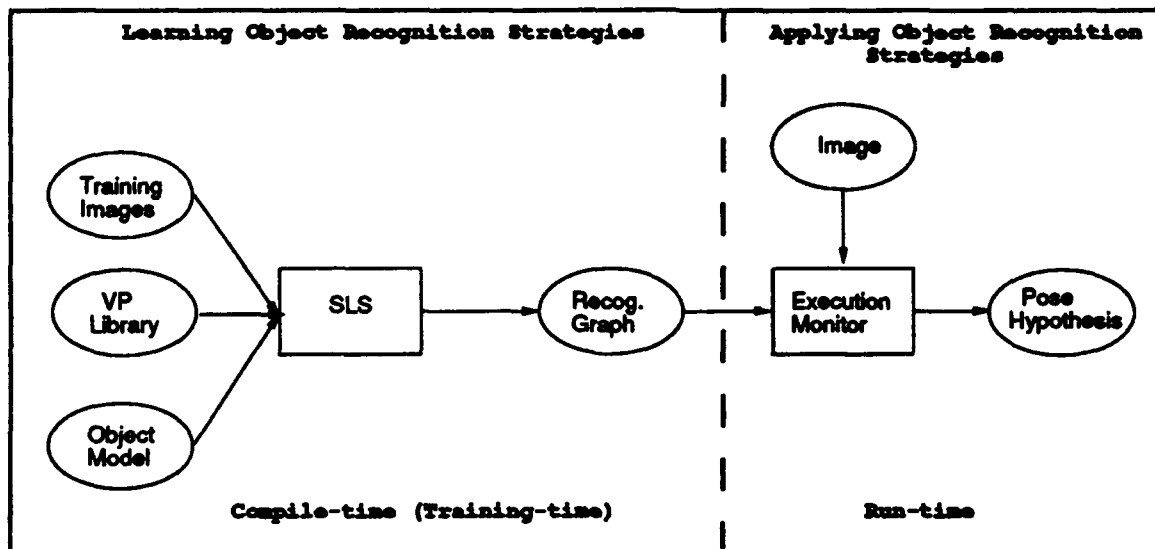


Figure 2: Top-level view of SLS architecture

Although TPs are described as transformation operators, the word 'transformation' should not be construed as implying a one-to-one mapping between old and new hypotheses. TPs can combine information from multiple hypotheses and may generate an arbitrary number of new hypotheses. Stereo line matching, for example, combines two image (2D) line hypotheses to generate a single world (3D) line hypothesis. In addition, TPs do not consume their arguments, so multiple TPs may be applied to a single hypothesis. Some readers may therefore find it helpful to think of TPs as procedures that generate new hypotheses from old hypotheses, rather than as a transformation operator.

3.1.2 Feature Measurement Procedures (FMPs)

Feature measurement procedures (FMPs) calculate features of hypotheses, such as orientations of planar surfaces and intensities of regions. During the recognition process, many properties of a hypothesis may remain uncalculated, so the set of known features describing a hypothesis is referred to as its *knowledge state*. Applying a FMP to one or more hypotheses computes a feature of those hypotheses, advancing them to new knowledge states. The number of knowledge states is finite, since continuous features are divided into discrete feature ranges.

3.1.3 Object Models

Many visual procedures, including various types of graph matching, require data from an object model to be compared with hypotheses extracted from the image. Since the object model does not change from image to image, SLS considers object model components to be compile-time parameters of visual procedures. When a strategy is trained, the object model,

supplied by the user, determines which visual procedures are enabled (either because they do not require model data, or because the necessary data is included in the model), and therefore which procedures can be used in the recognition strategy. For example, Beveridge's geometric model matcher [Beveridge and Riseman, 1992] matches 3D model lines to 2D data lines. If the object model includes a wire-frame shape description, the edges of the wire-frame become compile-time parameters to the model matcher specifying the 3D model lines to be matched. If, on the other hand, a wire-frame description is not included as part of the object model, then the geometric model matcher is not enabled and cannot be used as part of a recognition strategy.

3.2 Recognition Graphs

Interpretation strategies are represented in SLS as generalized multi-level decision trees called *recognition graphs* that direct both hypothesis formation and hypothesis verification, as shown in Figure 3. The premise behind the formalism is that object recognition is a series of small verification tasks interleaved with representational transformations. Recognition begins with trying to verify hypotheses at a low level of representation, separating to the extent possible hypotheses that are reliable from those that are not. Verified hypotheses (or at least, hypotheses that have not been rejected) are then transformed to a higher level of representation, where a new verification process takes place. The cycle of verification followed by transformation continues until 3D pose hypotheses are verified, or until all hypotheses have been rejected.

The structure of the recognition graph reflects the verification/transformation cycle. Each level of the recognition graph is a decision tree that controls hypothe-

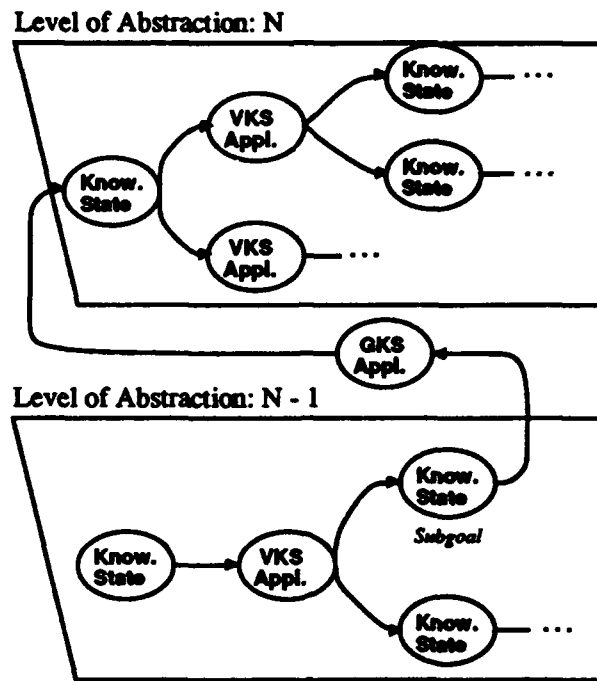


Figure 3: A recognition graph. Levels of the graph are decision trees that verify hypotheses using feature measurement procedures (FMPs). Hypotheses that reach a subgoal are transformed to the next level of representation by a transformation procedure (TP).

sis verification at one level of representation by invoking feature measurement procedures (FMPs) to gather support for or against each hypothesis. When a hypothesis is determined to be reliable within the decision tree, a transformation procedure (TP) transforms it to another level of representation, where the process repeats itself.

As defined in the field of operations research, decision trees are a form of state-space representation composed of alternating *choice states* and *chance states*. When searching for a path from the start state to a goal state, an agent is only allowed to choose where to go next from a choice state. If the current state is a chance state the next state is selected probabilistically³. The search process is therefore similar to using a game tree against a probabilistic opponent.

In SLS, the choice states are hypothesis knowledge states as represented by sets of hypothesis feature values. The choice to be made at each knowledge state is which FMP (if any) to execute next. Chance states in the tree represent FMP applications, where the chance concerns which value the FMP will return. Hypothesis verification is an alternating cycle in which the control

strategy selects which FMP to invoke next (i.e., which feature to compute), and the FMP probabilistically returns a feature value. Thus hypotheses advance from knowledge states to FMP application states and then on to new knowledge states. The cycle continues for each hypothesis until it reaches a subgoal state, indicating that it has been verified and should be transformed to a higher level of representation, or a failure state, indicating that the hypothesis is unreliable and should be rejected.

In general, SLS learns in advance which FMP to choose at each knowledge state in order to avoid making run-time control decisions. As a result, when SLS builds a recognition graph it leaves just one option at each choice node. Often, however, the readiness of a FMP to be executed cannot be determined until run-time, in which case SLS will leave several options at a choice node, sorted in order of desirability⁴. At run-time the system will choose the highest-ranking FMP that is ready to be executed.

3.3 Inference Algorithms

This paper will not repeat the inference algorithm by which SLS infers recognition graphs from training im-

³Operations research terminology is based on trees rather than spaces, so it refers to choice nodes and chance nodes rather than choice states and chance states, and to leaf nodes and root nodes rather than goal states and start states.

⁴This is just one of many complications that arise from multiple-argument visual procedures. In general, we will describe SLS as if all VPs took just one argument in order to keep the description brief; see Draper [1993] for a more complete description.

ages; interested readers are referred to [Draper et. al., 1993, Draper 1993, Draper et. al. 1992]. However, we will briefly mention some of its most important characteristics. First, SLS's inference algorithm is a syntactic, logic-based algorithm that makes no assumptions about the visual procedures or representations it is manipulating other than the declarative knowledge about the type of hypotheses that each takes as input and produces as output. As a result, although Table 3.3 shows some of the visual procedures and representations that have been used in SLS experiments to date, this list is by no means exclusive: SLS could just as easily manipulate generalized cylinders as wire-frame models, and probably any algorithm found in the Image Understanding literature could be included in a SLS strategy.

Second, SLS's learning algorithm tries to minimize the expected cost of recognition, subject to the constraint that a strategy must recognize every object instance in the training set. As a prerequisite for the minimization process, SLS estimates the expected cost of every visual procedure and the likelihood of each feature, based on information gained from the training images.

Finally, SLS's inference algorithm is strictly a generalization algorithm. SLS starts by learning a strategy for finding the target object in the first training image; it then generalizes this strategy to account for the second training image, and the third, and so on, until eventually the strategy is general enough to find the object in every training image. As it generalizes, each new strategy is guaranteed to be less likely to fail on a new image than the strategy it was generalized from, and this implies that SLS's strategies can be analyzed using techniques introduced by Valiant [1984] (see below).

4 Statistical Properties of Learned Strategies

As discussed in the introduction, the most immediate, practical problem with knowledge-directed vision is the time needed to construct knowledge bases, a problem that SLS solves by automatically learning recognition strategies. Another problem with hand-crafted strategies, however, is that even when they can be constructed, there is no way to ensure their validity, since their performance, in terms of accuracy and reliability, is unknown. SLS addresses this problem by estimating, for each recognition strategy it learns, 1) the expected cost of applying that strategy, and 2) a probabilistic bound on the likelihood of failure. Given the recognition graph formalism, predicting the expected cost is trivial; predicting robustness, on the other hand, requires more complex probabilistic reasoning.

4.1 Predicting Expected Cost

As was discussed in Section 3.2, recognition strategies are represented as multi-level decision trees called recognition graphs. Each level of a graph corresponds to one level of representation (e.g. points, lines, surfaces), and the decision tree at that level controls the order in which features are measured. Hypotheses with features that SLS has learned are reliable are then transformed to the next level of representation, while unreliable hypothesis are discarded.

Since SLS estimates the likelihood of each feature during training, as well as the expected cost of each feature measurement procedure (FMP), it is a straightforward procedure to estimate the cost of verifying a hypothesis at any level of representation, given a recognition graph (the equations can be found in [Draper et. al. 1992, Draper 1993]). Furthermore, since the cost of transforming hypotheses from one level to another, as well as the average number of hypotheses per level, can also be estimated from the training data, the total expected cost of recognition is easily obtained.

Experimentally, Draper [1993] describes three experiments in which the expected cost of a strategy, over twenty test images, was predicted to within four percent of its actual cost. In all three cases, the error was a slight overestimate of the cost, due to differences in the paging behavior of visual procedures during training and testing.

4.2 Predicting Robustness

A more difficult task is to predict the robustness of a strategy. Intuitively, a robust strategy is one that reliably recognizes objects in test images. For the sake of analysis, however, we will concentrate on the subproblem of how robustly a strategy generates goal-level hypotheses from images through chains of intermediate-level hypotheses⁵. For example, if the goal is to locate a building to within three feet of its actual position, what is the probability that at least one correct hypothesis will be generated when presented with a picture of the building? The analysis has to take into account the possibility of failure at any step in the process, as well as the redundancy in many strategies.

4.3 Assumptions

Any analysis of an algorithm must make certain assumptions about the data. In this case, the analysis rests on three assumptions about the knowledge base and training set:

1. **Deterministic VPs.** The behavior of a visual procedure is fully determined by the properties of

⁵SLS's learning algorithm is not a strict generalization algorithm when goal-level verification is included in the analysis.

Transformation Procedure	Input	Output	Ref
Moravec Operator	Image or ROI	2D Points	[Moravec 1981]
Line Extraction	Image or ROI	2D Lines	[Boldt and Weiss 1989]
Region Segmentation	Image or ROI	Region Set	[Beveridge, et. al. 1989]
Color Classification	Region Set	Region (sub)Set	[Duda and Hart 1973]
Polygonal Approx.	Region	2D Lines	
Parabola Fitting	Region	Parabola	
Pencil Grouping	2D Lines	Pencil	[Collins and Weiss 1990]
Vanishing Point Anal.	Pencil	3D Orientation	[Collins and Weiss 1990]
Trihedral Grouping.	2D Lines	Trihedral Jcts	
Trihedral Angle Anal.	Trihedral Jcts	3D Orientation	[Kanatani 1988]
Reprojection(1)	3D Orientation & Region	3D Plane	
Reprojection(2)	3D Orientation & 2D Points	3D Points	
Absolute Orientation	3D Points	Pose	[Horn 1987]
Subgraph Isomorphism	2D Lines	2D Lines	[Ullman 1976]
Point Resection	2D Points	Pose	[USGS]
Geometric Model Match	2D Lines	Pose	[Beveridge and Riseman 1992]

Table 1: The current library of transformation procedures (TPs) in SLS. SLS integrates procedures at a syntactic level, based solely on knowledge of the types of hypotheses a procedure takes as input and produces as output. As a result, new procedures can be easily added to its library.

its arguments.

2. **Knowledge base sufficiency.** Every object instance can be recognised by some sequence of VPs in the knowledge base. By definition, when this assumption is violated there is no good recognition strategy.
3. **Randomly selected training images.** Training images are drawn at random from the same image distribution as the test images. Although often violated in practice, this assumption provides the theoretical basis for predicting a strategy's performance on test images from its performance on training images.

4.4 PAC Analysis

A method for formally analyzing algorithms that generalizes from positive examples was introduced by Valiant as part of his work on *probably almost correct* (PAC) learning [Valiant 1984]. Valiant proved (based on earlier work by Chernoff) that the probability of fewer than S successes in n independent Bernoulli trials, each with probability h^{-1} or greater, is less than h^{-1} , where:

$$n \leq 2h(S + \ln h). \quad (1)$$

As an example of how Equation 1 might be used, Valiant considered the traditional problem of selecting marbles from an urn. Assuming S distinct colors of marbles, the probability that the $(n+1)$ th marble selected at random will be of a different color from all of its n predecessors is less than h^{-1} , by Equation 1. (Alert readers may notice that the probability of seeing a new color drops each time a new color is seen, but that it is always at least as high as the final probability, which is sufficient to satisfy the lemma. In effect,

the lemma overestimates the number of training samples needed by assuming only that the probability of seeing a new color on the first sample was at least as high as the probability on the last sample. The lemma applies because the probability of seeing a new color decreases monotonically.) Significantly, the probability bound h holds for *any* distribution of colors.

Valiant notes in his proof that h^{-1} is used in two separate probabilistic bounds. Qualitatively speaking, the first (call it h_1^{-1}) addresses the possibility that the randomly selected training samples may not be representative and therefore may not include a frequently occurring sample type. The second probability (call it h_2^{-1}) reflects the observation that if some colors are very rare, they will probably not be seen during training, even though there is a finite probability that they may occur during testing. It is these double probabilities that give probably almost correct learning its name: with probability h_1^{-1} , the learned concept or strategy accounts for all but h_2^{-1} of the samples in the underlying distribution, hence it will probably be almost correct. Moreover, there is no reason why h_1 has to be equal to h_2 . Nonetheless, we will follow Valiant in setting $h_1 = h_2$ and using Equation 1 (See [Kearns 1990] for a treatment that considers h_1 and h_2 separately.).

4.5 Analyzing Strategies

Valiant used Equation 1 as the foundation for a computational theory of machine learning [Valiant84], but we will use it for a much more modest purpose, namely estimating the robustness of recognition strategies. At

each step in its learning process, SLS has a strategy⁶ that is capable of recognizing every training instance seen so far. When a new training instance is presented, the current strategy is tested on it; if the strategy is already capable of recognizing the new object instance, then the current strategy is not changed, otherwise it is generalized to account for the new example. The situation is exactly analogous to Valiant's example of drawing balls from an urn, where the current strategy corresponds to the set of colors seen so far. As a result we can place a lower bound on the robustness of a strategy (i.e. an upper bound on the probability that it will fail) by counting the number of training instances and how often during training the strategy had to be generalized, and applying Equation 1.

Unfortunately, Equation 1 is not in a convenient form for determining the robustness of a strategy h from the number of training samples n and the number of failures during training S . Doing some algebra (and substituting m for $\frac{n}{2}$):

$$\begin{aligned} m &= h(S + \ln h) \\ e^m &= e^{Sh} e^{h \ln h} \\ &= e^{Sh} h^h \\ &= (e^S h)^h \\ (e^m)^{e^S} &= (e^S h)^{e^S h} \end{aligned}$$

Substituting c for $(e^m)^{e^S}$ and y for $(e^S h)$, we get an equation of the form $c = y^y$. Solving for y :

$$\begin{aligned} \ln c &= y \ln y \quad (2) \\ \ln \ln c &= \ln y + \ln \ln y \\ &\approx (1 + o(1)) \ln y \\ \ln y &\approx \frac{1}{1 + o(1)} \ln \ln c \end{aligned}$$

Substituting for $\ln y$ from Equation 2 yields:

$$\begin{aligned} \frac{\ln c}{y} &\approx \frac{1}{1 + o(1)} \ln \ln c \\ \ln c &\approx y \frac{1}{1 + o(1)} \ln \ln c \\ y &\approx (1 + o(1)) \frac{\ln c}{\ln \ln c} \end{aligned}$$

Resubstituting for c and y we get:

$$\begin{aligned} e^S h &\approx (1 + o(1)) \frac{\ln(e^m)^{e^S}}{\ln \ln(e^m)^{e^S}} \\ &\approx (1 + o(1)) \frac{e^S \ln e^m}{\ln(e^S \ln e^m)} \end{aligned}$$

⁶Actually, SLS maintains a set of potential strategies, and does not select the optimal one until after the last training image.

$$\begin{aligned} &\approx (1 + o(1)) \frac{e^S m}{\ln e^S + \ln \ln e^m} \\ &\approx (1 + o(1)) \frac{e^S m}{S + \ln m} \end{aligned}$$

Implying that:

$$h \approx (1 + o(1)) \frac{n}{2(S + \ln n - \ln 2)} \quad (3)$$

Equation 3 estimates a lower bound on the robustness of a strategy from the size of its training set and the number of generalizations during training, assuming only that $n > 2(S + \ln n - \ln 2) > 0$. In particular, it asserts that the probability of learning a strategy that fails more often than h^{-1} is less than h^{-1} .

5 conclusion

For small domains, knowledge-directed vision systems can recognize objects accurately and efficiently, in part because knowledge engineers are able to select the appropriate procedures and devise efficient control procedures for applying them. Unfortunately, as problem domains become more complex, it becomes impossible to hand-craft solutions to every recognition problem. SLS presents a solution to this knowledge engineering problem by having the system learn specialized recognition strategies from training images and object models, without the aid of a human knowledge engineer. SLS selects the most appropriate recognition techniques for a target object from a library of visual procedures, and builds a strategy that efficiently and effectively controls their application. Just as important, SLS does what most knowledge engineers were unable to do, namely analyze its strategies and predict their expected cost and robustness.

References

- [1] Beveridge, J.R., Griffith, J., Kohler, R.R., Hanson, A.R. and Riseman, E.M. "Segmenting Images Using Localized Histograms and Region Merging," *International Journal of Computer Vision*, 2:311-347 (1989).
- [2] Beveridge, J.R. and Riseman, E.M. "Can Too Much Perspective Spoil the View? A Case Study in 2D Affine and 3D Perspective Model Matching," *Proc. of the DARPA Image Understanding Workshop*, San Diego, CA., Jan. 1992, pp. 655-663.
- [3] Boldt, M., Weiss, R. and Riseman, E.M. "Token-Based Extraction of Straight Lines," *IEEE Trans. on Systems Man, and Cybernetics*, 19(6):1581-1594 (1989).
- [4] Collins, R.T. and Weiss, R.S. "Vanishing Point Calculation as a Statistical Inference on the Unit

- Sphere." *Proc. of the International Conference on Computer Vision*, Osaka, Japan, Dec., 1990, pp. 400-405.
- [5] Draper, B.A., Collins, R.T., Brolio, J. Hanson, A.R., and Riseman, E.M. "The Schema System," *International Journal of Computer Vision*, 2:209-250 (1989).
 - [6] Draper, B.A. and Riseman, E.M. "Learning Knowledge-Directed Visual Strategies," *Proc. of the DARPA Image Understanding Workshop*, San Diego, CA., Jan. 1992, pp. 933-940.
 - [7] Draper, B.A., Hanson, A.R., and Riseman, E.M. "Learning Blackboard-based Scheduling Algorithms for Computer Vision," to appear, *International Journal of Pattern Recognition and Artificial Intelligence*.
 - [8] Draper, B.A. "Learning Object Recognition Strategies," Ph.D. thesis, Univ. of Massachusetts, 1993.
 - [9] Duda, R.O., and Hart, P.E.. *Pattern Classification and Scene Analysis*. New York: John Wiley & Sons, 1973.
 - [10] Harvey, W.A., Diamond, M. and McKeown, D.M. "Tools for Acquiring Spatial and Functional Knowledge in Aerial Image Analysis", *Proc. of the DARPA Image Understanding Workshop*, San Diego, CA., Jan. 1992, pp. 857-873.
 - [11] Horn, B.K.P. "Closed-Form Solution of Absolute Orientation using Unit Quaternions," *Journal of the Optical Society of America*, 4(4):629-642 (1987).
 - [12] Hwang, V.S-S., Davis, L.S., and Matsuyama, T. "Hypothesis Integration in Image Understanding Systems," *Computer Vision, Graphics, and Image Processing*, 36:321-371 (1986).
 - [13] Kanatani, K. "Constraints on Length and Angle," *Computer Vision, Graphics, and Image Processing*, 41:28-42 (1988).
 - [14] Kearns, M.J. *The Computational Complexity of Machine Learning*. Cambridge, MA.: MIT Press, 1990.
 - [15] Matsuyama, T. "Expert Systems for Image Processing: Knowledge-Based Composition of Image Analysis Processes" *Computer Vision, Graphics, and Image Processing*, 48:22-49 (1989).
 - [16] McKeown, D.M. Jr., Harvey, W.A., and McDermott, J. "Rule-Based Interpretation of Aerial Imagery," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7(5):570-585 (1985).
 - [17] McKeown, D.M. Jr., Harvey, W.A., and Wixson, L.E. "Automating Knowledge Acquisition for Aerial Image Interpretation" *Computer Vision, Graphics, and Image Processing*, 46:37-81 (1989).
 - [18] Moravec, H.P. *Robot Rover Visual Navigation*. Ann Arbor, MI.: UMI Research Press, 1981.
 - [19] Ullman, J.R. "An Algorithm for Subgraph Isomorphism," *Journal of the ACM*, 23(1):31-42 (1976).
 - [20] Valiant, L.G. "A Theory of the Learnable," *Communications of the ACM*, 27(11):1134-1142 (1984).

Section XI

Active Vision Attention and Selective Perception

Detecting Activities

Ramprasad Polana and Randal Nelson

Department of Computer Science
University of Rochester
Rochester, New York 14627

Abstract

The recognition of repetitive movements characteristic of walking people, galloping horses, or flying birds is a routine function of the human visual system. It has been demonstrated that humans can recognize such activity solely on the basis of motion information. We present a novel computational approach for detecting such activities in real image sequences on the basis of the periodic nature of their signatures. The approach suggests a low-level feature based activity recognition mechanism. This contrasts with earlier model-based approaches for recognizing such activities.

1 Introduction

The motion recognition ability of the human visual system is remarkable. People are able to distinguish both highly structured motion, such as that produced by walking, running, swimming or flying birds, and more statistical patterns such as that due to blowing snow, flowing water or fluttering leaves. More subtle movement characteristics can be distinguished as well. For example, human observers can not only distinguish walking from other activities, but can also recognize a friend walking at a distance by his or her gait. Similar discrimination abilities have been observed in non-human animals. This biological use of motion probably reflects the fact that for certain tasks, visual motion provides more effective cues than other modes of visual perception. Motion is a particularly useful cue for certain types of recognition due to the fact that it is relatively easy to extract the motion field independent of illumination and shading of the image.

The classic demonstration of pure motion recognition by humans is provided by Moving Light Display experiments. In these experiments, reflective pads are attached to the joints of a person and his or her movements are filmed against a black background so that only the light reflected off the pads is visible. When people are shown these images they dismiss single frames as meaningless dot patterns, but they recognize the sequential presentation of them as walking, running or jumping etc. Subjects can also identify the actor's gender and even iden-

tify the actor if known to them [Johansson, 1973]. For certain recognition tasks, as illustrated above, motion alone is sufficient.

Duplication of some of these motion recognition abilities in machine systems would be useful in a number of applications. One area is in automated surveillance. Motion detection via image differencing can be used for intruder detection; however such systems are subject to false alarms, especially in outdoor environments, since the system is triggered by anything that moves, whether it be a human, a dog, or a tree blown by the wind. Motion recognition techniques, both of the discrete and textural variety have the potential to disambiguate the motions of different origin. Another application is in industrial monitoring. Many manufacturing operations involve a long sequence of simple operations each performed repeatedly and at high speed by a specialized mechanism at a particular location. It should be possible to set up one or more fixed cameras that cover the area of interest, and to characterize the allowed motions in each region of the image(s).

A useful first step in recognizing motion from gray-level image sequences is to classify motions according to the spatial and temporal uniformity they exhibit. We define *temporal textures* to be the motion patterns of indeterminate spatial and temporal extent, *activities* to be motion patterns which are temporally periodic but are limited in spatial extent, and *motion events* to be isolated simple motions that do not exhibit any temporal or spatial repetition. Examples of temporal textures include wind blown trees or grass, turbulent flow in cloud patterns, ripples on water, the motion of a flock of birds etc. Examples of activities are walking, running, or flying individual, rotating or reciprocating machinery, etc. Examples of motion events are isolated instances of opening a door, starting of a car, throwing a ball etc.

It turns out that temporal textures can be effectively treated with statistical techniques analogous to those used in gray-level texture discrimination. A previous paper [Polana and Nelson, 1992] describes this. Activities and motion events, on the other hand, are more discretely structured, and techniques similar to those used in static object recognition would be expected to be useful in their classification. Since different sorts of techniques must be used to distinguish the different sorts of motion, it would be useful to have a method for mak-

ing a preliminary classification of the motions present in an image. In this paper, we describe a robust method for detecting and localizing periodic activities, including ones, such as walking or flying, that involve simultaneous translation of the actor. The method is based on frequency domain analysis of a image in which low-level motion information has been used to isolate and track likely locations of activity. The method also suggests a way of using low-level structural features to classify activities once they have been detected.

2 Related Work

Although motion plays an important role in biological recognition tasks, motion recognition in general, has received little attention in the literature compared to the volume of work on static object recognition. Most computational motion work in motion in fact, has been concerned with various aspects of the structure-from-motion problem. There is a large body of psychophysical literature addressing the perception of motion, most of it concerned with primitive percepts. A modest amount of this work addresses more complicated motion recognition issues [Johansson, 1973, Cutting, 1981, Hoffman and Flinchbaugh, 1982, Hildreth and Koch, 1987], but the models and descriptions have typically not been implemented. Various computational models of temporal structure, have been proposed (e.g. [Chun, 1986, Feldman, 1988]) but much of this work is at a fairly high level of abstraction, and has not actually been applied to visual motion recognition except in rather artificial tests.

Goddard [1989] considers recognizing event sequences from Moving Light Display (MLD) images. His work addresses the representation of motion event sequences and their recognition assuming certain invariant image features. His input consists of the joint angles and angular velocities computed from the motion of the dots in the light displays. The joint angles and angular velocities are invariant to rotation in the image plane, scale and translation. A challenging part in computing these invariants is to recover the connectivity of the individual dots (by body parts) in the MLD images. A domain independent approach to this problem is given by Rashid. Rashid [Rashid, 1980, O'Rourke and Badler, 1980] considered the computational interpretation of moving light displays, particularly in the context of gait determination. This work emphasized rather high-level symbolic models of temporal sequences, an approach made possible by the discrete nature of the moving light displays. The results were quite sensitive to discrete errors and thus highly dependent on the ability to solve the correspondence problem and accurately track joint and limb positions. This severely limits the general applicability of the method.

Anderson et al. [Anderson et al., 1985] describe a method of change detection for surveillance applications based on the spectral energy in a temporal difference image. This was not generalized to other motion features or more sophisticated recognition. Gould and Shah [1989] represent motion characteristics of moving objects by recording the important events in their trajectory.

They propose the use of the resulting *trajectory primal sketch* in a motion recognition system. Koller, Heinze and Nagel [1991] developed a system that tracks moving vehicles and characterizes their trajectory segments in terms of natural language concepts.

A few studies have considered highly specific aspects of motion recognition computationally. Pentland [Pentland and Mase, 1989] considered lip reading, and implemented a system that could recognize spoken digits with 70%-90% accuracy over 5 speakers. The system required the location of the lips to be entered by hand, and depended on an explicitly constructed lip model. Some temporal pattern recognition work has been done in the context of speech processing [Juang and Rabiner, 1985, Tank and Hopfield, 1987, Elaman, 1988]. But the applicability of the techniques to motion recognition has not been considered.

3 Activity Detection

Activities involve a regularly repeating sequence of motion events. If we consider an image sequence as a spatio-temporal solid with two spatial dimensions and one time dimension, then repeated activity tends to give rise to periodic or semi-periodic gray level signals along smooth curves in the image solid. We refer to these curves as *reference curves*. If these curves could be identified and samples extracted along them over several cycles, then frequency domain techniques could be used in order to judge the degree of periodicity.

Two important cases are stationary activities, and activities that result in a more or less uniform translation of the actor, i.e. locomotory activities. If the activity is stationary, the reference curves are lines parallel to the temporal dimension. For example, a circularly rotating ring gives rise to a temporally periodic signal at every pixel. In the case of uniform translation, the curves are straight lines at some angle that depends on the velocity. For general translation and perspective projection, the lines associated with a given actor form a bundle with a common intersection, the vanishing point. For many practical situations, however, the vanishing point is far enough removed that the lines can be considered to be effectively parallel.

Consider, for example the case of walking. This is an example of a non-stationary activity; that is, if we attach a reference point to the person walking, that point does not remain at one location in the image. If the person is walking with constant velocity, however, and is not too close to the camera, then reference point moves across the image on a path composed of a constant velocity component modulated by whatever periodic motion the reference point undergoes. Thus, if we know the average velocity of the person over several cycles, we can compute the spatio-temporal line of motion along which the periodicity can be observed. If the person moves with average velocity (u, v) the spatio-temporal line of motion will be determined by the equations $(x, y) = (u, v) * t + (x_0, y_0)$, where (x, y) is the position of the object in space at time t and (x_0, y_0) is the position at time zero. This applies to any object undergoing constant velocity locomotion.

3.1 Periodicity Detection

From Fourier theory we know that any periodic signal can be decomposed into a fundamental and harmonics. That is, we can consider the energy of a periodic signal to be concentrated at frequencies which are integral multiples of some fundamental frequency. This implies that if we compute the discrete Fourier transform of a sampled periodic signal, we will observe peaks at the fundamental frequency and its harmonics. Hence, in theory, the periodicity of a signal can be detected by obtaining its Fourier transform and checking whether all the energy in the spectrum is contained in a fundamental frequency and its integral multiples.

The real-world signals, however are seldom perfectly periodic. In the case of signals arising from activity in image sequences, disturbances can arise from errors in the uniform translation assumption, varying background and lighting behind a locomoting actor, and other sources. In addition, for computational purposes, we need to truncate the signal at some finite length which may not be an exact integral multiple of its period. Nevertheless, the frequency defined by the highest amplitude often represents the fundamental frequency of the signal. Hence we can get an idea of the periodicity in a signal by summing the energy at the highest amplitude frequency and its multiples, and comparing that quantity to the energy at the remaining frequencies. In practice, since peaks in a Fourier transform tend to be slightly broadened for a variety of reasons, including the finite length of the sample, we define the periodicity measure p_f of a signal f as a normalized difference of the sum of the power spectrum values at the highest amplitude frequency and its multiples, and the sum of the power spectrum values at the frequencies halfway between. That is,

$$p_f = (\sum_i F_{iw} - \sum_i F_{(iw+w/2)}) / (\sum_i F_{iw} + \sum_i F_{iw+w/2})$$

where F is the energy spectrum of the signal f and w is the frequency corresponding to the highest amplitude in the energy spectrum.

The measure is normalized with respect to the total energy at the frequencies of interest so that it is one for a completely periodic signal and zero for a flat spectrum. In general, if a signal consists of frequencies other than one single fundamental and its multiples, its periodicity measure will be low.

Because the signal along any given reference curve in the image solid may be ambiguous, we need a way of combining periodicity measures of a number of signals from reference curves associated with the same actor. The simplest idea would be simply to sum the power spectra of the various signals, and apply the periodicity measure to the resultant curve. Unfortunately, this does not work, primarily because, although there is a fair amount of energy at the fundamental frequency, and quite a few signals in which high periodicity is present, there are also a lot of samples where the periodicity is not evident, or which appear periodic at some other frequency. The net affect, is that all this energy at other frequencies can swamp the main signal if they are combined additively. What does work, is a form of non-maximum

suppression, where the periodicity measure is obtained for each power spectrum separately. Each frequency w is then assigned a value equal to the sum of the periodicity measures P_w from all the signals whose highest amplitude occurred at that frequency. The result is the same as suppressing all but the maximum frequency in each transform, weighting each by the periodicity measure of the signal, and summing them. The maximum value of this combined signal is taken as the fundamental frequency, and the associated periodicity measure is the average of the periodicity measures of the contributing signals.

Thus, the periodicity measure P for an entire image sequence is defined as

$$P = \max_w (P_w / n_w)$$

where n_w and P_w are the number of pixels at which the highest amplitude frequency is w and the sum of the periodicity measures at those pixels respectively.

Finally, in order to apply the technique to real data, we need a way of extracting reference curves and the associated signals from an image sequence. In the following, we assumed that any activity that existed in the data would be either stationary, or locomotory in a manner that produced an overall translatory motion. We also assumed that there was at most one actor in the scene, though a certain amount of background motion could be tolerated. The first assumption turns out not to be too restrictive - a large number of natural periodic activities fit into one of the two categories. The second can be relaxed with some additional preprocessing. The first step is to identify locations in the scene where movement of any sort is occurring. This is done by computing the normal flow magnitude at each pixel between each successive pair of frames using a spatio-temporal derivative method. Those pixels at which significant motion is present are marked, and the centroid of the marked pixels computed in each frame. The mean velocity (if any) of the actor is then computed by fitting a linear trajectory to the sequence of centroids. This is where the one-actor assumption comes into play. If several actors were present, simple clustering techniques could be used to isolate the regions in the scene corresponding to different activities. The reference curves were taken the lines in the spatio-temporal solid parallel to that generated by the linear-fitted trajectory of the centroid. Signals were extracted along these curves, and those that displayed significant spread over a period of at least half as long as the signal were selected for processing. This had the effect of eliminating the need to process regions in which no motion occurred, as well as regions affected only by an occasional blip.

3.2 Experiments

We ran experiments on four different activities, and a number of non-periodic motions. The sequences were first recorded on video and then digitized later with suitable temporal sampling so that at least four cycles of the activity were captured in 128 frames. Following is a description of each activity and the conditions under which they were digitized.

- **Walk:** A person walking across a room viewed in profile. Six sequences of 128 frames of size 128x128 pixels were obtained. Half the sequences contained one person and the other half a second.
- **Exercise:** A person performing jumping jacks. Four sequences of 128 frames of 128x128 pixels, two each of two different people.
- **Swing:** A person swinging viewed from the side. Six sequences of 128 frames of 128x128 pixels, three each of two different people.
- **Frog:** A toy frog simulating swimming activity viewed from above. Four sequences of 128 frames of 64x256 pixels.
- **Nonperiodic:** Various sequences taken from television shows and live outdoor shots: splashing water, closeup of crowd at a political rally, a plane flying overhead, a robot hand picking up and manipulating objects (2 sequences), the input to an eye tracker (eyeball movements), leaves fluttering in the wind, turbulent flow in a stream. In all, 8 sequences of 128 frames of 128x128 pixels.

The swing and exercise activities were shot outdoors and contained background motion as well. Among the periodic activities, a single sequence of uniform rotation is included as well. Sample images of periodic activities are shown in figures 1.

The periodicity measures computed using the above algorithm are plotted for all 20 periodic and all 8 non-periodic sequences in figure 2. As is evident from the graphs and the projected scatter plot, the technique separates complex periodic from non-periodic motion nicely. The requirement that an empirically determined threshold be used is not a great drawback in this case, nor is it particularly surprising, since even the intuitive notion of periodic activity falls on a continuum. Is the motion of a branch waving somewhat irregularly in the wind periodic or non-periodic? Here, we classified it as non-periodic, but it had one of the higher periodicity measures, as might be expected.

4 Discussion

The periodicity detection method we described satisfies the several desirable invariances. It is invariant to image illumination, contrast, translation, rotation and scale. It is also invariant to the magnitude of locomotory motion and the speed of the activity. It is also fairly robust with respect to small changes in viewing angle. The periodicity measure does not depend on the number of pixels involved in the activity. If desired, a restriction on the minimum number of pixels can be imposed so that only activities of a minimum size can be recognized. The swing and exercise sequences were taken outdoors where there is a small amount of background motion. This comprises not only moving trees and plants, but also moving people and occasional crossing of a car. That periodicity can be detected even in this case demonstrates that the technique is reasonably tolerant of background clutter and an occasional disturbance. The technique also provides a method for localizing activity in the scene

by back-projecting the reference curves having high periodicity measures into the image solid.

So far we have assumed that the actors giving rise to the activity move with constant velocity along linear paths. The case of nonlinearly moving objects can be handled by tracking the object of interest given a coarse estimate of its initial location and velocity. This would generate reference curves that were not straight lines. We have already demonstrated the usefulness of the centroid of motion for computing the velocity of linearly moving objects. It could also be used for tracking the actors moving on more complex trajectories. Use of the motion centroid can be unreliable in estimating the centroid of the object if the shape of the object changes as it moves. In this case use of a prediction and correction mechanism using past values over a sufficiently long period can help.

The detection scheme also assumes that there is only one activity in the scene except for some background clutter. If there are multiple activities in the scene, this detection technique can still be applied provided the activities can be spatially isolated so that they do not interfere with each other. In this case they can be segmented using the motion information and later tracked separately. Even an occasional crossing of different activities can be tolerated as long as the regions can be separated again later.

The complexity of detection is proportional to the number of pixels involved in the activity. About half the work is computing the fast Fourier transforms at each of the pixels. Most of the rest of the time is occupied by the motion detection process. The detection procedure currently runs on an SGI machine using four processors and it takes approximately 15 seconds to process a 128 frame sequence of 128x128 images.

4.1 Recognition of Activities

The first stage in recognizing an activity is to detect that an activity exists, and localize it in the scene. This paper has described a technique for accomplishing this. Future work will utilize information computed in the detection stage for recognition and classification of specific activities. The detection scheme utilizes only the magnitude of the Fourier transform to obtain the periodicity measure. The phase of the Fourier transform is also computed at each location in the image and we propose to use this information along with other low-level information in the image, for recognition. For example, walking can be described as a sequence of motion events regularly occurring at each spatial location. The cycle of motion events at different spatial locations in the image have a fixed phase difference. These phase differences are valuable in characterizing the activities.

5 Conclusion

We have described a method of activity detection. This technique uses a periodicity measure on gray-level signals extracted along spatio-temporal reference curves. We have illustrated the technique using real-world examples of activities, and shown that it robustly detects complex periodic activities, while excluding non-periodic motion.

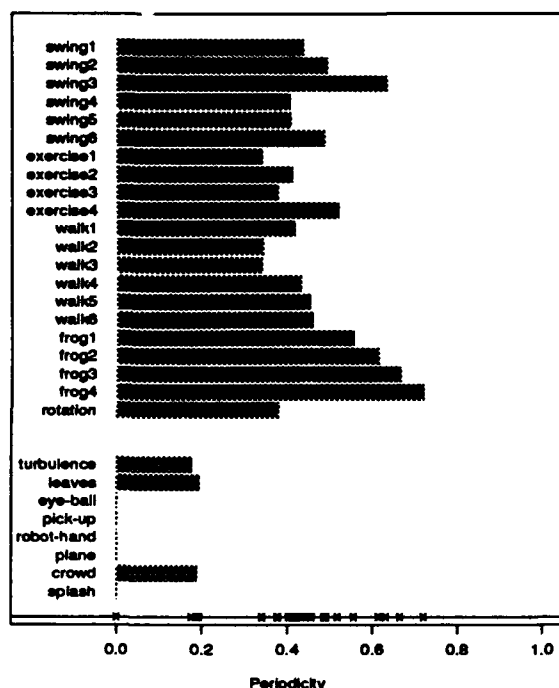


Figure 2: Periodicity measure for Periodic and Nonperiodic sequences

We proposed a technique to recognize these activities using the detection scheme described here. It is not clear how much the periodicity alone is useful for recognition but we believe the phase information is valuable for activity recognition. Future work will concentrate on the development of robust phase features that can be used in conjunction with previously developed motion and gray-level features to classify activities.

References

- [Anderson *et al.*, 1985] C. H. Anderson, P. J. Burt, and G. S. van der Wal. Change detection and tracking using pyramid transform techniques. In *Proc. SPIE Conference on Intelligent Robots and Computer Vision*, pages 300-305, 1985.
- [Chun, 1986] H.W. Chun. A representation for temporal sequence and duration in massively parallel networks: Exploiting link connections. In *Proc. AAAI*, 1986.
- [Cutting, 1981] J.E. Cutting. Six tenets for event perception. *Cognition*, pages 71-78, 1981.
- [Elaman, 1988] J.E. Elaman. Finding structure in time. Technical Report 8801, Center for Research in Language, Univ. of California, San Diego, 1988.
- [Feldman, 1988] J.E. Feldman. Time, space and form in vision. Technical Report 244, University of Rochester, Computer Science Department, 1988.
- [Goddard, 1989] N.H. Goddard. Representing and recognizing event sequences. In *Proc. AAAI Workshop on Neural Architectures for Computer Vision*, 1989.
- [Gould and Shah, 1989] K. Gould and M. Shah. The trajectory primal sketch: A multi-scale scheme for representing motion characteristics. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 79-85, 1989.
- [Hildreth and Koch, 1987] E.C. Hildreth and C. Koch. The analysis of visual motion from computational theory to neural mechanisms. *Annual Review of Neuroscience*, 1987.
- [Hoffman and Flinchbaugh, 1982] D.D. Hoffman and B.E. Flinchbaugh. The interpretation of biological motion. *Biological Cybernetics*, pages 195-204, 1982.
- [Johansson, 1973] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14:201-211, 1973.
- [Juang and Rabiner, 1985] B.H. Juang and L.R. Rabiner. Mixture autoregressive hidden markov models for speech signals. *IEEE Trans. Acoustics, Speech and Signal Processing*, 6:1404-1413, 1985.
- [Koller *et al.*, 1991] D. Koller, N. Heinze, and H.-H. Nagel. Algorithmic characterization of vehicle trajectories from image sequences of motion verbs. In *Proc. of IEEE Computer Vision and Pattern Recognition*, pages 90-95, 1991.
- [O'Rourke and Badler, 1980] J. O'Rourke and N.I. Badler. Model-based image analysis of human motion using constraint propagation. *PAMI*, 3(4):522-537, 1980.
- [Pentland and Mase, 1989] A. Pentland and K. Mase. Lip reading: Automatic visual recognition of spoken words. Technical Report 117, M.I.T. Media Lab Vision Science, 1989.
- [Polana and Nelson, 1992] R. Polana and R.C. Nelson. Temporal texture recognition. In *Proc. of CVPR*, pages 129-134, 1992.
- [Rashid, 1980] R.F. Rashid. *LIGHTS: A System for Interpretation of Moving Light Displays*. PhD thesis, Computer Science Dept, University of Rochester, 1980.
- [Tank and Hopfield, 1987] D. W. Tank and J. J. Hopfield. Concentrating information in time: analog neural networks with applications to speech recognition problems. In *Proceedings of the First International Conference on Neural Networks*, pages 455-468, 1987.

Acknowledgements

This work is supported by contracts NSF IRI-9010692 and AFOSR 91-0288.

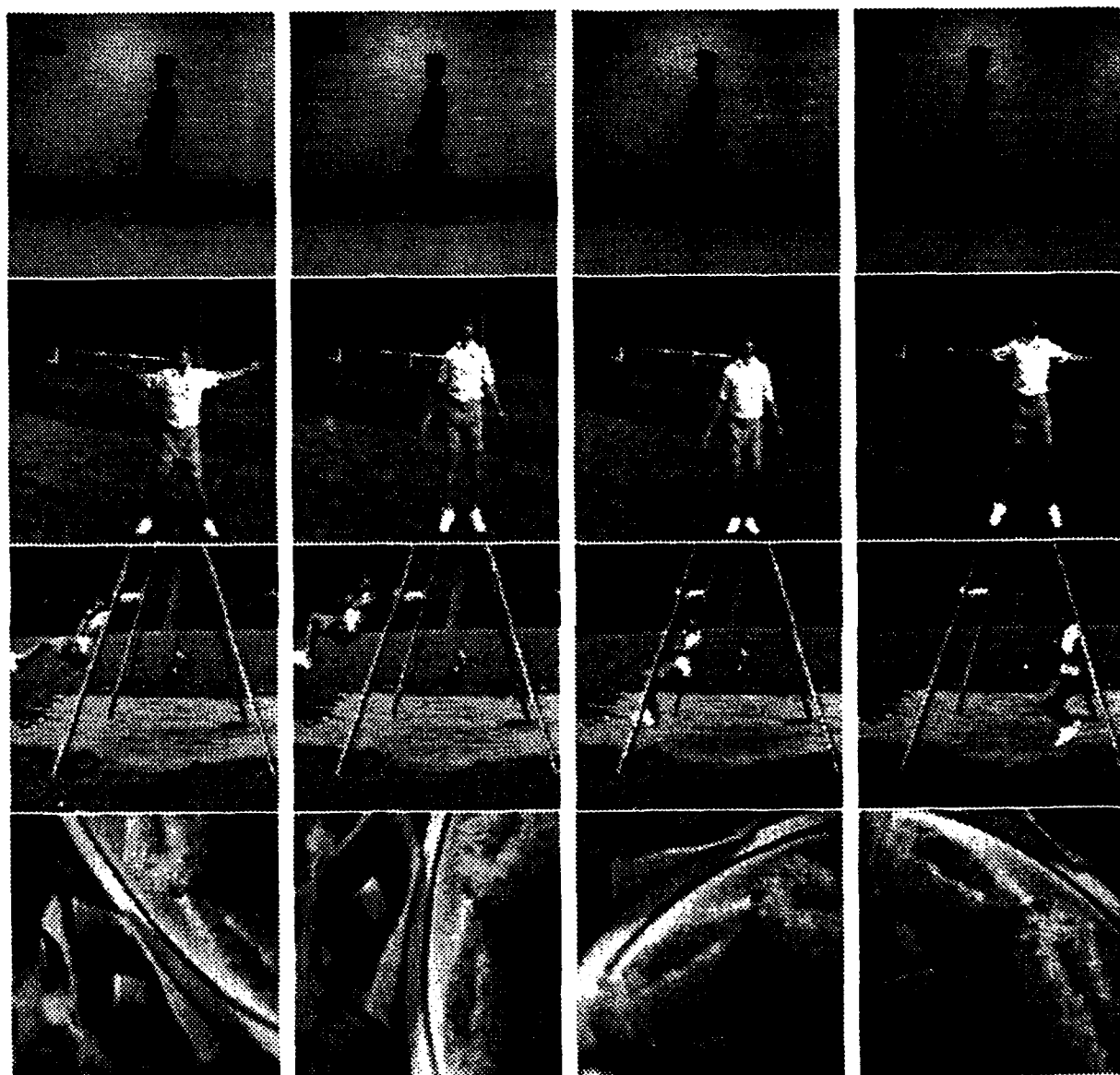


Figure 1: Sample images from periodic sequences: walk, exercise, swing and rotation

Studying Control of Selective Perception With T-world and TEA

Raymond D. Rimey and Christopher M. Brown*

The University of Rochester

Computer Science Department

Rochester, New York 14627-0226

rimey@cs.rochester.edu and brown@cs.rochester.edu

Abstract

We hypothesize that selective perception allows more accurate solutions to visual tasks to be found in less wall-clock time than non-selective techniques. The best way to assess the practical truth of this hypothesis is by studying, designing and building complete vision systems — the issues are fundamentally systems issues. On the other hand, special-case systems are not convincing: we present the *T-world problem* as an abstraction of an interesting class of real-world vision problems. T-world has enough structure to support basic study of fundamental tradeoffs inherent in selective computer perception. Our complete system is called TEA-1: it is a purposive and sufficing vision system that solves a version of the T-world problem. TEA-1 is a fully implemented system, and extensive experiments in the laboratory and simulation have explored the key factors that make the selective perception approach appealing, analyzing how each factor affects the overall performance when solving a set of automatically generated (in simulation) T-world domains and tasks.

1 Selective Perception

1.1 General Concepts

Purposive vision. A purposive vision system works to achieve a goal (i.e. solve a visual task) in minimal wall-clock time. Goal-directed operation can make the system fast by limiting the amount of data processed and by limiting the extent to which that data is processed.

Sufficing vision. Sufficing vision is the use of (usually simple, cheap, and general) vision modules whose output is ambiguous unless considered in a known context.

It is essentially impossible to design a vision module that performs well in all possible contexts. However, it is quite possible that through partial visual analysis or prior knowledge, the vision system has some information about the situation in the scene and the contexts

in which objects appear or are expected to appear. The idea of sufficing vision is that vision modules are designed for and only executed in such contexts. For example, when looking for the carrots at a dinner table it may be sufficient to look for a big blob of orange and then check the orange things are roughly elongated. In another situation it may be sufficient simply to look for a big blob of orange. Two things are required in order for sufficing vision to work. First, a system is needed that establishes contexts and uses them to specify exactly what vision modules to run. Second, a large repertoire of flexible vision modules is needed.

Historically, vision modules have been engineered to produce relatively high-level outputs, ones humans can reason about. Sufficing vision systems may be less transparent: A human may find it difficult to understand or specify exactly what a vision module in a sufficing vision system is really doing and why, since the context may not be known to the human and the significance of the extracted information within that context may not be obvious. While in general it may be difficult to design and integrate such vision modules, there are two causes for optimism. First, learning techniques may be able to tune and select modules in the context of the whole system, even though humans can not easily specify the modules explicitly and *a priori*. Second, one aspect of the sufficing vision idea is that existing (relatively simple) vision modules may be more useful than they may seem, when intelligently applied and interpreted within specific contexts.

Selective perception. A selective perception system is purposive and sufficing.

Control of selective perception. The general problem that we are interested in is to control (select actions and make decisions in) a computer vision system that has a repertoire of actions (sufficing vision modules) such that the system operates in a purposive manner.

We make the following general assumptions: The computer vision system has (high-level) knowledge about a domain and a set of tasks. A repertoire of actions is available that provides many different ways to obtain imperfect information about a scene. Characteristics (reliability, time needed) of the actions are known. The computer vision system has a pointable vision sensor, requiring that it make decisions about moving the vision sensor.

*This material is based on work supported by DARPA Contract MDA972-92-J-1012. The Government has certain rights in this material.

The tasks the system must solve are in one sense very simple. If a complete symbolic description of the scene were available, a subset of that information directly determines the solution to a task. The problem is that (a) the system must try to retrieve information about each member of the subset one at a time, (b) the scene is complex so it is difficult to locate the desired information, and (c) only imperfect information can be obtained. The visual actions are parameterized, providing information of varying quality. The control problem, crudely described as deciding "what to do next", really consists of several problems: what to look for next, where to look next, and how to look for things.

1.2 Hypotheses

Our goal is to prove or at least support the following hypotheses.

A (high-level) computer vision system (with limited resources and working in complex environments) that is purposive and sufficing is better than one that is not, meaning that it solves tasks in less wall-clock time.

Bayesian and decision theoretic techniques offer a sound formal basis for control of a purposive and sufficing computer vision system.

A control algorithm based on a carefully designed evaluation function with lookahead can implement purposive and sufficing vision.

The performance of a purposive and sufficing vision system depends on the amount of "structure" there is in a scene, meaning the various ways that objects may be grouped in a scene and the spatial relationships between objects and groups.

A sufficing computer vision system requires a large repertoire of actions whose performance characteristics are controlled by a set of parameters.

1.3 Fundamental Contributions

We believe that the best way to address many of our hypotheses is by studying, designing and building complete vision systems — the issues are fundamentally systems issues. The contributions of our work so far are summarized below.

We define the T-world problem, a simple class of vision problems that still contains many of the key factors motivating the selective perception approach. We believe T-world is an adequate problem for easily studying some of the basic issues in selective computer perception, and that the T-world problem can be mapped to a variety of real-world applications [Rimey, 1993]. We have worked with the abstract T-world problem in static and dynamic scene simulations and a real-world application (dinner table scenes) in the lab.

We present the TEA-1 system, an example of a purposive and sufficing vision system that solves a version of the T-world problem. Analysis and experiments are provided that explore the advantages of the TEA-1 system's purposive and sufficing behavior on the T-world problem. We explore the key factors that make the selective perception approach appealing, analyzing how each factor affects the overall performance when solving a set of automatically generated (in simulation) T-world do-

main and tasks. One crucial factor is the amount and nature of structure in the organization of objects in the scene.

TEA-1 shows one reasonable way to design a computer vision system using Bayes nets (aka influence diagrams) and decision theory. Other computer vision and robot systems have been built that use decision theoretic techniques, Bayes nets, Dempster-Shafer or similar modeling techniques (see Section 5), usually in the lower-level capacity of supporting sensor fusion for single object recognition or building environmental descriptions. TEA-1 addresses high-level computer vision, is one of only a few systems that consider how to control an actively pointable sensor, and is the first to emphasize purposive and sufficing control in high-level vision. Since TEA-1 is a *fully* implemented system, we have been able to perform extensive experiments in simulation and in the lab, and to analyze factors that affect selective perception — these are fundamentally systems issues — using *complete* runs on a *large number* of scenes.

There are many approaches to control in a selective perception system ranging from brute-force and heuristic search through hand-crafted evaluation functions to a formal planning system. Our work explores this spectrum of choices, studying and experimenting with some of the choices and exploring the issues in control and how each choice deals with those issues.

2 The T-world Problem

This section defines the T-world problem, a formalization of some key problem characteristics that can be exploited by a selective perception system.

A scene. A scene consists of many objects within a large two-dimensional rectangular area. Each object has a location (for its centroid), rectangular dimensions, a type, and a set of properties. Each property has a set of possible values. There may be any number of objects in the scene. Objects may overlap each other, but this does not affect the performance of a visual action (see below). The objects in the scene may be organized into a set of mutually exclusive groups, and groups may have subgroups, subsubgroups, etc. Subgroup structure is determined by the domain rules (see below).

The sensor. The sensor, called a camera, is a fixed-size rectangular window that is in the plane of the scene and is much smaller than the extent of the scene. The window may be moved (by an camera movement action, see below) to any location in the scene, specified by the coordinates for the center of the window in a two-dimensional world coordinate system. The window defines the camera's field of view. A fixed-size rectangular window, called the "fovea", is much smaller than the field of view's size, and may be moved around inside the field of view.

A low spatial-resolution image that covers the entire field of view is available and is called the "peripheral image". A high spatial-resolution image that covers the fovea is available and is called the "foveal image". (Alternatively, a resolution pyramid of images may be used.)

A domain. A "domain" consists of a set of scene types and a set of probabilistic rules for each scene type

that specifies the number, type, location (and grouping structure) and properties of objects in a scene.

A task. A task is defined as determining the value of a task variable, which is a (probabilistic) function of a subset of the number, type, location and properties of objects in the scene.

Camera movement actions. Given a specified location in the scene (in the two-dimensional world coordinate system) a "camera movement action" moves the camera so it is centered on the specified location. This action always moves the camera exactly to that location. The time to execute a camera movement action is a function of the distance moved.

Visual actions. Visual actions try to obtain information about the portion of the scene visible inside the field of view (*i.e.* from image data). There is a large collection of visual actions designed to obtain many different types of information. We currently use two types of visual action: one tries to detect a specific type of object in an image, and the other tries to obtain the value of a specified property of a specified object in an image.

The behavior of an action depends on whether the target object is truly in the field of view or not, the true type, location and properties of the target object and of all the other objects in the field of view. An action may have a precondition that must be satisfied before the action can be executed.

The performance of an action is a function of several parameters, which must be specified for each action: the image resolution (currently either foveal or peripheral resolution, and generally a level in a resolution pyramid), the image area to process, and the length of time to process a unit of image data. Note that several actions may have the same purpose, but different performance characteristics. The time to execute a visual action is a function of the specified parameters.

The problem. Given a scene from an identified T-world domain and a specified T-world task, the problem is to sequentially collect evidence from the scene to support a decision about the answer to the task, with a desired level of confidence, so that the total wall-clock time for executing the actions is minimized. Solving the problem involves the following general steps: decide what action to execute next, execute that action, incorporate the results from that action, decide on the answer to the task, and decide whether to gather more evidence or to stop.

A set of programs has been written that allows TEA-1 to analyze scenes and solve tasks in a simulated T-world (in addition to a version that runs in the lab). One program simulates an instance of T-world (scene, camera, actions, *etc.*) as specified by a database of rules and models. Another program automatically generates the database files that specify new instances of T-world domains, and scenes and tasks for each domain. The same program automatically generates the knowledge representation structures used by the TEA-1 system.

3 TEA-1: A Decision-Theoretic Solution for Control of Selective Perception in the T-world Problem

TEA-1 is an implemented, compact, flexible, selective computer vision system, which solves a version of the T-world problem and has a solid foundation of well-established formalisms — Bayesian statistics and decision theory. TEA-1 uses Bayes nets for representation and a cost and benefit analysis extending over action sequences to decide which visual or non-visual action to perform next. We believe TEA-1's current design provides a general software tool sufficient to study a variety of basic issues in high-level and low-level selective analysis and behavior in computer perception.

A probabilistic knowledge representation is appropriate for a selective system, and Bayes net and Dempster-Shafer approaches are two obvious alternatives. We choose the Bayes net approach because it is flexible and easy to use, and works well for the variety of tasks and domains we have in mind. We developed a version of Bayes nets, called a composite Bayes net, which consists of domain-specific knowledge (including geometrical) and a specification of the desired task. The composite net includes a new application of Bayes nets to represent relative object locations and geometric relations. A task is specified by a net that makes explicit the relation of evidence needed to accomplish a specific perceptual task to the components of the domain-dependent knowledge representation.

We have used generic, easily-tuned, sufficing vision utilities (histograms, Hough transforms) from our software library. These sufficing algorithms are in general simple and fragile; in a known context they are simple and robust. Our goal is to be able to use intelligently whatever visual operators are at hand. The control system can apply a vision module in a very specific spatial or semantic context, knowing how the context affects the performance of that module.

TEA-1's design assumes all the details in the T-world definition in Section 2. TEA-1 programs can transparently run either with a T-world simulator providing input and accepting output or in the laboratory (for a dinner table domain).

More details about TEA-1 are available in [Rimey and Brown, 1992a, 1992b, 1992c, 1992d, Rimey, 1993], including the various decision making algorithms, example runs of the system, and other experimental results.

4 Factors Affecting the Performance of Selective Perception

We are currently analyzing the relationship of several key factors to the overall performance of a selective perception system, using T-world and TEA-1: (1) automatically generate a large number of simulated T-world domains, scenes, and tasks; (2) run TEA-1 on the generated scenes and tasks; and (3) compute the average solution time over all scenes for each task. This approach lets us show how each factor affects the average solution time. Factors falling in four categories are be-

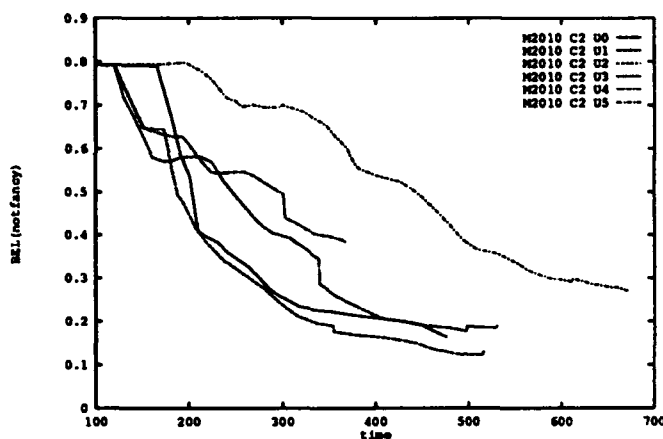


Figure 1: Plots of belief over time for various action evaluation functions. Belief: that a table setting is not fancy when it actually is, averaged over 10 scenes generated by the T-world simulator. U5 is a constant evaluation, or random choice of operation, and sophistication increases from U4 to U0. Better performance means both that the final belief value is lower and that it gets lower faster.

ing analyzed: control method, scene structure, systems parameters, and parameterizable actions [Rimey, 1993].

4.1 Control Method

Our early work explored a variety of evaluation functions (and related issues) for deciding what camera movements to make and what visual actions to execute. For example, Figure 1 shows the performance of TEA-1 solving a table setting task with six different action evaluation functions, holding other factors constant. We have also explored several different evaluation functions for making camera movement decisions, and have compared the evaluation function approach with a state-space search of all possible action sequences [Rimey and Brown, 1992a, 1992c].

Many fundamental questions remain, however, as to the most effective control and cost/benefit evaluation mechanisms. For example, the original TEA-1 design used a $V_{rate} = value/cost$ measure, though some (camera movement calculations) used $V_{imm} = value - cost$. Calibration between value and cost is necessary in either method. Computing the value of an action as V_{imm} emphasizes finding the best single action to perform now, but maximizing V_{rate} ensures the fastest improvement over time, which is also an important consideration. The latest TEA-1 design maximizes a combination of V_{rate} and V_{imm} (and action value is now based on the expected value of sample information rather than on average mutual information). Preliminary results using T-world and TEA-1 regarding these questions are encouraging. We are currently changing some more details of the implementation to be more consistent, so tighter results can be obtained for comparisons, and so more extensive comparisons can be made.

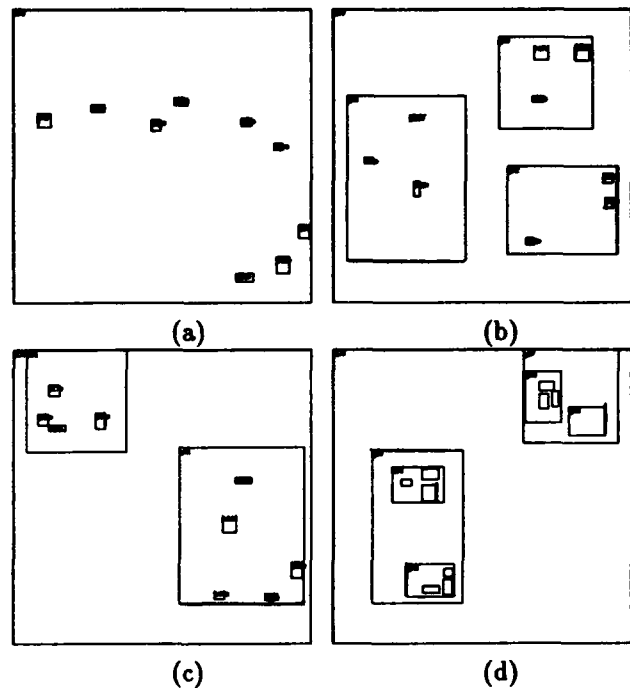


Figure 2: Some examples of how a scene of nine objects could be structured. The smallest squares are objects. The other squares depict groups or subgroups. (a) No grouping. (b) Three groups of three objects. (c) Two rather dense groups, with five and four objects in each. (d) A subgroup structure.

4.2 Scene Structure

Several aspects of scene structure can have a significant impact on performance, mainly because geometric relations define contexts in a scene, which help locate things in the scene and thus help obtain more accurate information more cheaply. See [Rimey and Brown, 1992a, 1992b, 1992d] for an example in a dinner table domain that shows how a cup's expected area gets narrower and how an actual cup detection action performs after each of the table, plate, and napkin have been located (in that order).

We are currently studying several specific scene-structure factors via simulated T-world domains, such as: number of groups and number of subgroup levels that objects are organized into, average number of geometric relations between objects, and shape (and type) of geometric relation distributions between objects. For example, Figure 2 shows several different ways that a scene of nine objects could be structured.

Another factor that we classify under scene structure is whether it is inherently easier to detect and obtain properties of some objects than others. For example, certain large objects (like runways) may be easier to find than small objects (like service vehicles), while highly constraining the location (or properties) of smaller objects. Wixson demonstrated that the efficiency of actively searching for a specified target object in a room can be improved by a factor to 2 to 8 when a related intermediate object is located first [Wixson, 1992].

The T-world problem contains significantly more opportunity for and kinds of scene structure than Wixson's object search problem, which contains only the simple "look near" relation. The solution to a T-world problem can involve several types of information about several objects, rather than simply detecting one object. It will be interesting to compare the performance gains obtained by using relations with more than one object to those obtained by using relations with only one object.

Some of the task complexity factors that T-world enables us to analyze experimentally are: the minimum number of (independent) scene features needed to solve the task; loosening the independence restriction; and whether all features have the same impact on the task, or some features contribute more information to the task's solution.

4.3 Parameterizable Actions

Selective perception (and qualitative vision) tightly couples high-level control with the low-level vision modules, specifically so the vision modules can be asked to provide only the minimal information needed to solve the current task. How much and what type of flexibility must exist in the vision modules, and how does the control system mesh with such a repertoire of very flexible vision modules (*e.g.* with several parameters)? This question has not received much attention: most vision modules have been designed to recover as complete a description as possible, to support traditional vision tasks like single object recognition or scene property reconstruction.

The value of some vision algorithms is affected by parameters (depth of search, spatial resolution, iteration count, annealing schedule, *etc.*) that change operation cost. Our decision-theoretic formalism can be extended to a continuous range of benefit/cost choices that will allow the control of monotonic or anytime algorithms whose results get better with longer running times.

4.4 System Parameters

The system parameters category includes: performance model of a visual action (a table of probabilities), relative costs of visual and non-visual actions, size of the camera's field of view, size of the fovea, relative speed of computation (in the multiprocessor version of TEA-1). Figure 3 shows the effect of varying the cost of a camera movement, which is generally to stretch the performance curve over time [Rimey and Brown, 1992a]. (Wixson discusses the effect of some similar parameters on an object search task in [Wixson, 1992].)

Varying some system parameters will change the system's overall pattern of behavior, meaning the best sequence of actions to execute, which can produce interesting effects and raises interesting issues. For example, making camera movements more expensive means that more time is spent analyzing more of the things visible at each camera fixation.

4.4.1 Cycles

The combination of cheap camera movements and "anytime" visual actions [Dean and Wellman, 1991] could encourage cyclic fixation and analysis patterns to

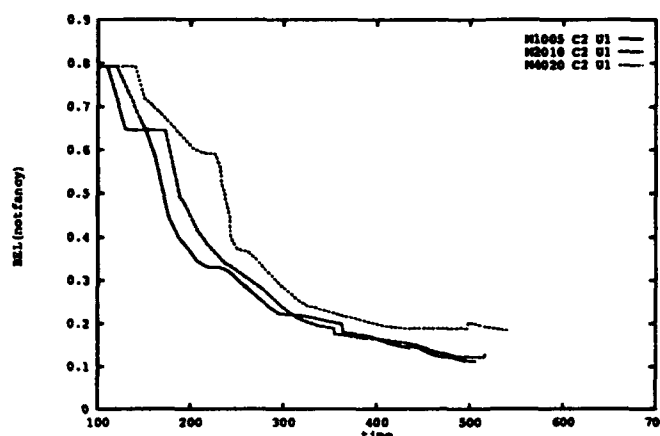


Figure 3: Performance for a dinner table task as the camera movement cost is varied.

emerge. T-world circumvents knowledge engineering and other practical difficulties in experiments with complex scenes, so we can study these issues.

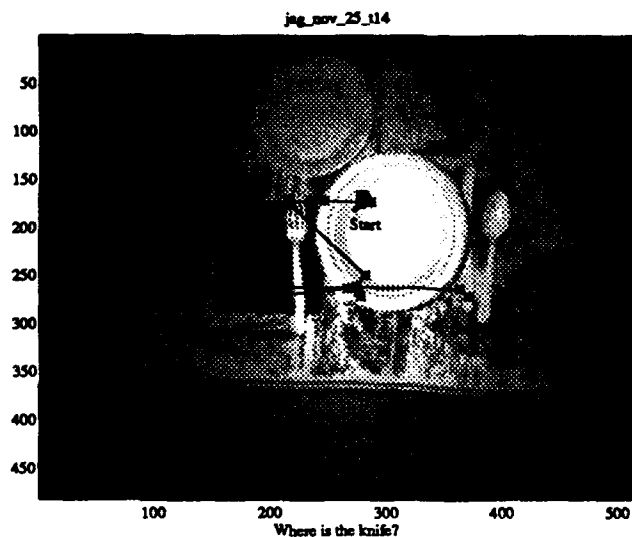
With cheap vision, humans may not use their innate powers of representation and memory and may prefer just to update short-term memory. This strategy seems to be found in humans [Ballard *et al.*, 1993] in repetitive sequential hand-eye tasks. On the other hand, human eye fixations during even simple tasks clearly show evidence of rational sequential control ([Yarbus, 1967], and see Figure 4). Further, vision is expensive when peripheral vision is reduced, when there are distractors, noise, low contrast, *etc.* Humans *do* manage their visual resources, even for static scenes, and their management strategies are open to investigation through several avenues. We are hopeful that we can relate decision-theoretic control to human performance by using modern eye-, head-, and hand-tracking technology to observe humans performing T-world tasks.

5 Related Work

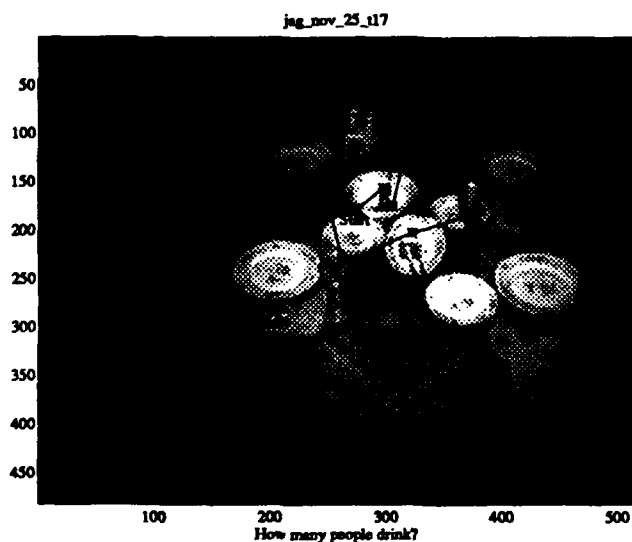
Extensive references can be found in our other papers [Rimey and Brown, 1992a, 1992b, 1992c, 1992d, Rimey, 1993]. Our work on task-based vision is most directly comparable to [Hutchinson and Kak, 1989], which put a carefully designed version of model-based hypothesis verification vision into a Dempster-Shafer setting, and the related [Wu and Cameron, 1990, Hager, 1990]. Additional key computer vision and robotics applications of decision theory are [Levitt *et al.*, 1989] and [Dean *et al.*, 1990].

Acknowledgements

Peter von Kaenel helped build parts of the T-world simulator and several vision modules and visual actions. Tim Becker improved modularity of the T-world/TEA-1 system, and parallelized several parts of it. Martin Jagersand obtained the eye movement recordings in Figure 4, using software being developed by Jeff Pelz.



(a)



(b)

Figure 4: Eye-fixations in visual tasks. (a) Where is the knife? (S habitually places knife to left of plate). (b) How many people drink? (S uses plates somehow).

References

- [Ballard *et al.*, 1993] Ballard, D.; Hayhoe, M.; Li, F.; and Whitehead, S. 1993. Hand-eye coordination during sequential tasks. *Phil. Trans. R. Soc. Lond. B* 335 (in press).
- [Dean and Wellman, 1991] Dean, T. L. and Wellman, M. P. 1991. *Planning and Control*. Morgan Kaufmann.
- [Dean *et al.*, 1990] Dean, T.; Camus, T.; and Kirman, J. 1990. Sequential decision making for active perception. In *Proceedings: DARPA Image Understanding Workshop*. 889-894.
- [Hager, 1990] Hager, G. D. 1990. *Task-Directed Sensor Fusion and Planning: A Computational Approach*. Kluwer Academic.
- [Hutchinson and Kak, 1989] Hutchinson, S. A. and Kak, A. C. 1989. Planning sensing strategies in robot work cell with multi-sensor capabilities. *IEEE Journal of Robotics and Automation* 5(6):765-783.
- [Levitt *et al.*, 1989] Levitt, T.; Binford, T.; Ettinger, G.; and Gelband, P. 1989. Probability-based control for computer vision. In *Proceedings: DARPA Image Understanding Workshop*. 355-369.
- [Rimey and Brown, 1992a] Rimey, R. D. and Brown, C. M. 1992a. Control of selective perception using Bayes nets and decision theory. *International Journal of Computer Vision*. Submitted.
- [Rimey and Brown, 1992b] Rimey, R. D. and Brown, C. M. 1992b. Task-oriented vision with multiple Bayes nets. In Blake, A. and Yuille, A., editors 1992b, *Active Vision*. MIT Press. 217-236.
- [Rimey and Brown, 1992c] Rimey, R. D. and Brown, C. M. 1992c. Task-specific utilities in a general Bayes net vision system. In *Proceedings: IEEE Conference on Computer Vision and Pattern Recognition*. 142-147.
- [Rimey and Brown, 1992d] Rimey, R. D. and Brown, C. M. 1992d. Where to look next using a Bayes net: Incorporating geometric relations. In *Proceedings: European Conference on Computer Vision*. 542-550.
- [Rimey, 1993] Rimey, R. D. 1993. *Control of Selective Perception*. Ph.D. Dissertation, Department of Computer Science, University of Rochester. Forthcoming.
- [Wixson, 1992] Wixson, L. 1992. Exploiting world structure to search for objects efficiently. Technical Report 434, Department of Computer Science, University of Rochester.
- [Wu and Cameron, 1990] Wu, H. L. and Cameron, A. 1990. A Bayesian decision theoretic approach for adaptive goal-directed sensing. In *Proceedings: International Conference on Computer Vision*. 563-567.
- [Yarbus, 1967] Yarbus, A. 1967. *Eye Movements and Vision*. Plenum Press, New York.

Exploratory Active Vision

Jean-Yves Hervé and Yiannis Aloimonos

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742

Abstract

What is an active observer? In the literature, it is considered to be an observer capable of controlling its sensory apparatus, and thus its image acquisition process. Activities can be movement, tracking, zooming, focusing, etc. when the sensory modality is vision. But whatever the particular activity may be, its effect is to alter the visual input so that it becomes easier to extract from it the visual quantities of interest.

If we examine various activities more closely, we find that they amount to the addition of a motion field to the visual input. In cases such as movement and tracking (whether smooth pursuit or saccade tracking), this field is a rigid motion field corresponding to a 3D motion. It was shown ([Aloim88b]) that equipping the observer with any movement capability makes various visual recovery problems easier and gives them uniqueness properties as well.

But what happens at the algorithmic level? Are all activities—rigid motions in this case—the same as regards stability issues? In other words, can the active observer explore the space of all activities in order to discover the one that provides the most stable solution? This is the problem we investigate in this paper.

1. Introduction

1.1 Overview

From a theoretical point of view, an active observer needs to perform various partial recovery tasks in order to take the appropriate action, and the question we pose is “what activity (motion) will provide the most robust solution regarding the structure of the scene in view?” While addressing this problem, we present as a by-product of our analysis a solution that unifies shape from shading, shape from texture, and shape from motion. No optical flow or correspondence are used in our approach.

We need to emphasize that this work is of a theoretical nature. By no means do we suggest that an active observer needs to perform a complete reconstruction of the scene in view. On the contrary, our recent studies indicate that this is not necessary. In this paper, we simply show that an active observer which needs to recover information about the structure of the scene in view can employ many activities. We prove that one of the activities in this set will provide the most stable solution (in an algorithmic sense), and we show how it can be found. This, in turn, suggests that it is fruitful to study exploratory active perception, whatever the sensory modality or the activity may be.

1.2 Motivation for this work

One of the main topics of research in modern computer vision is the “shape from x” problem. Following the paradigm introduced by David Marr ([Marr82]), numerous models and algorithms have been proposed that attempt to explain or mimic the behavior of *modules* observed in the human or animal visual system and that recover the geometry of an observed scene, using cues such as shading, texture, motion, stereo, etc. The goal of Marr’s paradigm was not only to provide models to explain animal vision, but also to offer a methodology and tools for designing artificial visual systems that could be used in robotics tasks, visual navigation tasks in particular. By clearly separating the visual module from the motion planning module, it allows them to be studied and developed independently from each other.

The great appeal of the Marr paradigm resides in its generality: regardless of the task to be performed (and, therefore, regardless of the type of robot in question), the interface between sensing and planning is provided by the depth map computed by the visual module. In this respect, the static (open-loop) position-based, look-and-move controller (using the classification proposed by [Sande83]) represented in Figure 1, corresponding for example to the hand-eye system of Tsai and Lentz ([TsaiL89]), can be considered to be the general model of a visual reconstruction-based system. Although a dynamic (closed-loop) version of this model is theoretically

possible (and was discussed in [Sande83]), it has virtually never been realized, mainly due to the prohibitive computational cost of visual reconstruction (calibration) processes. What is generally presented as closed-loop control is therefore a sequence of error-reducing runs of an open-loop controller.

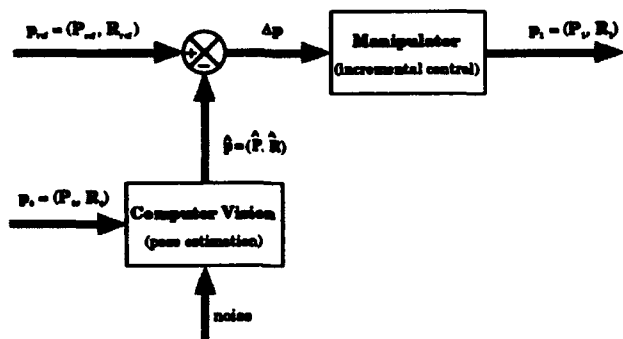


Figure 1 — Position-based open loop control

At the other end of the robotics problem, current work on robot motion planning, even when assuming the existence of a visual sensor (whether in a theoretical analysis, as in [Chati85], or in an implemented system such as the ones described in [Weiss84] and [Fedde89]), completely ignores the sensing process. In particular, it is always assumed that the visual module can provide reliable information, regardless of the activities of the robot observer¹.

The results presented in [Aloim88b] clearly demonstrate that this is not the case; some activities (motions) of the observer can be shown to make the visual algorithms more stable. The question that naturally comes next is: Can such a good activity be determined *a priori*? This question is immediately followed by: How can this choice of a "good" activity (in terms of visual computations) be incorporated into the motion planning process (in terms of the task)?

1.3 Shape from x modules

The goal of this paper is not to demonstrate that extraction of information about the structure of the visual environment can be facilitated by the employment of an active observer. This has been demonstrated in a previous paper [Aloim88b]. We simply demonstrate that among the infinite class of activities that an observer can employ, there exists one which is optimal in the sense of stability—i.e., when the observer employs this activity, information about shape is recovered in the most robust manner. Since information about shape is hidden in shading, texture, and motion, in a sense our work unifies the following shape from x modules:

- **shape from shading**, which is concerned with the

¹ Work dealing with uncertainty in motion planning is no exception to this observation, since at one point or another, perfect information about the scene or the motion is needed in all work published so far.

smooth variations of light intensity over the image ([Gibso50], [Horn77]).

- **shape from texture**, which is concerned with the variation of distributions of image discontinuities or elementary discontinuity patterns ([Gibso50]).
- **shape from motion** (or structure from motion, as it is more often called), which attempts to extract depth information from the displacement of image features ([TsaiH84]) or the modification of the image intensity ([HornS81]) resulting from the motion of the observer or of objects in the scene.

1.4 The active observer

The concept of Active Perception was introduced in [Bajcs86] and further analyzed in [Aloim88b]. An active observer, when engaged in an activity, modifies the constraints underlying a given phenomenon (and the equations describing them) and thus creates new information that helps to eliminate ambiguities and make the solution easier to find and, often, more reliable (that is, more robust). It was shown in [Aloim88b] that classical, difficult, or even ill-posed vision problems can be made simpler if the observer accomplishes some activity chosen from the space of possible activities such as movement, tracking, focusing, eye convergence, touch, etc. Active vision can also be seen as a technique for the integration of shape from x modules, for example, the shape from texture and shape from shading modules ([Aloim89]).

2. Shape Recovery

2.1 Notation

The observer considered here is a monocular optic system (camera), which we represent by a classical pin-hole model (Figure 2).

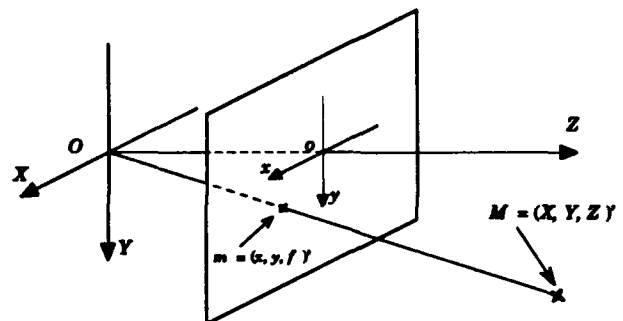


Figure 2 — Pinhole model of the observer

The 3D space is referenced to a viewer-based coordinate system $\mathcal{R} = (O, OX, OY, OZ)$, where O and OZ are the optical center and the optical axis of the camera respectively, and OZ intersects the image plane orthogonally at o , with $d(Oo) = f$, the focal length of the camera. OX and OY are defined so as to be parallel to the axes of the image plane.

Let $M = (X, Y, Z)'$ be a point of the 3D world. M projects onto the image plane as a point $m = (x, y, f)'$. Since we use a pinhole model, the following relation holds between M and m :

$$m = \frac{f}{Z} M. \quad (1)$$

This allows us, if we consider m as a function of M , to compute the Jacobian matrix

$$\frac{\partial m}{\partial M} = \frac{1}{Z} D \quad \text{where} \quad D = \begin{pmatrix} f & 0 & -x \\ 0 & f & -y \\ 0 & 0 & 0 \end{pmatrix}. \quad (2)$$

2.2 Motion of the observer

The observer is moving with a known rigid motion composed of a translation $T = (t_1, t_2, t_3)'$ and a rotation $\omega = (\omega_1, \omega_2, \omega_3)'$. A point M in the observed scene is therefore seen as moving with the apparent velocity $v = -\omega \times M - T$. In order to simplify the expressions to follow, we define Ω to be the antisymmetric matrix associated with the linear operator $\omega \times$:

$$\Omega = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}.$$

Using these equations, we can now express the apparent motion of world point M , as seen by the observer, as $\dot{M} = -\Omega \cdot M - T$. The motion of M 's projection on the image plane, m , defines the *motion flow*

$$\dot{m} = \frac{\partial m}{\partial M} \cdot \frac{dM}{dt} = \frac{1}{Z} D \cdot (-\Omega \cdot M - T). \quad (3)$$

2.3 The optical flow equation

For a given motion of the observer and two consecutive images of a scene or surface, $I(t_0)$ and $I(t_0 + \Delta t)$, with Δt assumed to be small enough to justify a differential approximation, we want to reconstruct the shape of the surface, i.e. recover $Z(x, y)$.

An apparently simple idea would be to compute the optical flow \dot{m} and report it in (3) to obtain Z for each point in the image. The optical flow constraint traditionally used to relate the unknown optic flow to the image intensity data is the one proposed by [HornS81]:

$$\frac{dI}{dt} = \nabla I \cdot \dot{m} + \frac{\partial I}{\partial t} = 0. \quad (4)$$

This equation merely states that the images of M at times t and $t + dt$, $m(t)$ and $m(t + dt)$ respectively, can be considered to have the same intensity. Although this simple model somewhat lacks realism—for example, it does not take into account specular phenomena—it has nevertheless been adopted by most researchers in this domain, partly due to its intuitiveness and to its analytic simplicity, but also due to the fact that more complex equations do not perform significantly better. We can, however, formulate the following remarks before going any further in our analysis:

- Equation (4) does not give the optical flow, but the *normal flow*, i.e. the component of the flow along the direction of ∇I . Providing we add extra constraints (smoothness), regularization techniques can give us a solution ([HornS81]); unfortunately, they do not handle discontinuities well. Theories of discontinuous regularization have been proposed ([Shulm88]), but have only been partially applied to the problem of optical flow computation ([Shulm89]).
- A direct exploitation of (4) requires the computation of ∇I , which is known to be an ill-posed problem ([Poggi85]). Furthermore, it restricts the applicability of the module to the case of smooth variations of the intensity (i.e. shape from shading).
- Even assuming the flow can be calculated, the criterion optimized by the depth map thus obtained remains unnatural (smoothness of the flow) or unclear. What does it imply about the flow when we minimize norms of its derivatives? What does the depth map corresponding to such a flow look like?

It seems clear that an important flaw in flow-based methods is that they require pointwise calculations of derivatives and other operators, while the data we are given (intensities) are locally inaccurate. On the other hand, the reliability of averages computed on portions of the image is quite satisfactory. This is why we introduce linear features ([Amari86]) in our theory.

2.4 Linear features

A linear feature is a triple $LF_i = (\mu_i, \varphi_i, \Sigma_i)$ where

- Σ_i is an image window.
- The *measuring function* μ_i is a differentiable function defined over Σ_i .
- φ_i is defined as a moment over Σ_i :

$$\varphi_i = \iint_{\Sigma_i} \mu_i I ds = \iint_{\Sigma_i} \mu_i(x, y) I(x, y) dx dy.$$

It can be shown (see Appendix) that if we adopt (4) as a model of intensity variation the time derivative of φ_i takes the following form:

$$\dot{\varphi}_i = \frac{d}{dt} \left(\iint_{\Sigma_i} \mu_i I ds \right) = \iint_{\Sigma_i} I \nabla \mu_i \cdot \dot{m} ds. \quad (5)$$

If we now replace \dot{m} in (5) by the expression (3) for the motion flow, we obtain the following equation, where Z is the unknown:

$$\begin{aligned} \dot{\varphi}_i = & - \iint_{\Sigma_i} \frac{1}{Z} I \nabla \mu_i \cdot (D \cdot T) ds \\ & - \frac{1}{f} \iint_{\Sigma_i} I \nabla \mu_i \cdot (D \cdot \Omega \cdot m) ds. \end{aligned} \quad (6)$$

2.5 Model-based solution of the equation

The right hand term of equation (6) is known; what we need now is a way to get $1/Z$ out of the integral

sign. This can be accomplished if we model it as a linear combination of differentiable functions (for example, monomials, Gabor functions, or Fourier components):

$$\frac{1}{Z} = \sum_{k=1}^m \alpha_k \psi_k(x, y). \quad (7)$$

Replacing $1/Z$ by its model (7) makes the vector of coefficients $\mathbf{A} = (\alpha_k)_{k=1,m}$ the unknown of the final equation:

$$\sum_{k=1}^m \left[\iint_{\Sigma_l} (I \psi_k \nabla \mu_l \cdot \mathbf{D} \cdot \mathbf{T}) ds \right] \alpha_k = -\dot{\varphi}_l - \frac{1}{f} \iint_{\Sigma_l} (I \nabla \mu_l \cdot \mathbf{D} \cdot \boldsymbol{\Omega} \cdot \mathbf{m}) ds. \quad (8)$$

or, in a more compact form:

$$(\mathbf{S}_l \cdot \mathbf{T})' \cdot \mathbf{A} = r_l$$

where \mathbf{S}_l is an $m \times 3$ matrix and r_l is a scalar.

2.6 The least squares solution

Each linear feature $(LF)_l$, $l=1,n$ provides a linear equation similar to (8). A least squares minimization exploiting this system of equations gives us the following equation:

$$\underbrace{\left[\sum_{l=1}^n (\mathbf{S}_l \cdot \mathbf{T} \cdot \mathbf{T} \cdot \mathbf{S}_l') \right]}_{\mathbf{Q}} \cdot \mathbf{A} = \underbrace{\left(\sum_{l=1}^n \mathbf{S}_l \cdot \mathbf{T} r_l \right)}_{\mathbf{R}} \quad (9)$$

2.7 Comments on equation (9)

We do not need to compute any pointwise derivatives. In particular, the term ∇I does not appear in our equations. This allows discontinuities to occur in the image: the method does not require any particular property of the intensity map.

The only derivatives we need to calculate in order to recover Z are the $\dot{f}_l \simeq [\varphi_l(t + \Delta t) - \varphi_l(t)] / \Delta t$, where we have to keep in mind that φ_l is obtained by integration over the image window Σ_l and is thus more stable and reliable than values at individual points.

We are not restricted to the use of the optical flow equation (4); every linear feature gives us a linear equation. Choosing a big enough linear feature vector, we can get a solution by Hough transform or least squares approximation.

The criterion optimized by the solution is observer-based; for a given shape model, the α_k 's we compute are such that the expected intensity variation (which we can predict, given $I(x, y, t_0)$ over Σ_l , and the motion parameters) is closest to what is actually observed in the image.

Satisfactory results obtained by similar techniques (using a simplified version of (8)) have been reported in earlier papers ([Aloim89], [Aloim88a]) and will therefore not be reproduced here, since *reconstructing* the depth map is not the main focus of this work.

3. Search for a Best Activity

3.1 Optimization in the space of activities

We have seen that an active observer can create new information and will be able to solve problems which were ill-posed for a passive observer. Still, common sense tells us that not all activities are equally good; some will not actually help eliminate ambiguities. If we want to choose a "good" activity, we need to answer the following questions:

- is there a goodness criterion valid in the space of all possible activities?
- if such a criterion exists, can we find an activity which optimizes it?

If the problem treated here were purely mathematical, one could think of the accuracy of the computed shape model (i.e. its distance from the actual $Z(x, y)$) as a good estimate of the activity. Unfortunately, we are dealing with the real world: data are noisy and the reliability of the camera calibration and of the motion parameters is questionable. Under such conditions, the only criterion that makes sense is the stability of the solution under perturbation of the input and parameters: we want to pick an activity which will optimize the stability of the solution calculated. But can we?

3.2 Stability of the least squares solution

We first comment that the rotational part of the motion has no effect on the stability of equation (9), only on the recomputed depth map, which is not the main focus of the work presented here. Our search for a best activity will therefore be reduced to an optimization in the two-dimensional space of translation directions.

An obvious condition that \mathbf{T} has to satisfy is for \mathbf{Q} to be regular, i.e. $\text{Det } \mathbf{Q} \neq 0$. But this is not enough; not only do we want \mathbf{Q} to be regular, we want it to be "as regular as possible". In other words, we want \mathbf{Q} to behave well during the numerical solution of the equation (for example, small pivots should not be encountered in a triangulation of the matrix). This imposes conditions on the eigenvalues of \mathbf{Q} . If \mathbf{Q} has small eigenvalues (at this stage, "small" is purposely vague, meaning, roughly "anything that will bring intermediate results close to the roundoff error"), it will not be singular, but it will be ill-conditioned and the computed solution will be unstable.

Let $\lambda_1, \dots, \lambda_m$ be \mathbf{Q} 's eigenvalues listed in increasing order (since \mathbf{Q} is obtained by least squares minimization, the λ_i 's are real and positive). The condition number of \mathbf{Q} is defined as the ratio

$$C = \frac{\lambda_m}{\lambda_1}$$

i.e. the ratio of the largest to the smallest eigenvalue of \mathbf{Q} . If the condition number is infinite, \mathbf{Q} is singular; if C is large, \mathbf{Q} is ill-conditioned. The best \mathbf{T} we could choose would be one which would minimize the condition number. This problem, however, is difficult since we cannot

give an analytic expression for \mathcal{C} . Instead, we propose to minimize the following function:

$$\begin{aligned}\mathcal{G}(\mathbf{Q}) &= \ln \left[\frac{\text{Det } \mathbf{Q}}{(\text{Tr } \mathbf{Q})^m} \right] = \ln(\text{Det } \mathbf{Q}) - m \cdot \ln(\text{Tr } \mathbf{Q}) \\ &= \ln(\lambda_1 \lambda_2 \dots \lambda_m) - m \cdot \ln(\lambda_1 + \lambda_2 + \dots + \lambda_m).\end{aligned}$$

This choice of \mathcal{G} may seem somewhat arbitrary and needs to be justified here:

- First, \mathcal{G} is well defined, since the determinant and the trace of \mathbf{Q} are invariant under a diagonalization process, and the fraction cannot be equal to zero (since $\lambda_m \geq \lambda_1 \geq 0$), unless \mathbf{Q} is equal to the null matrix.
- The logarithmic term is used here to simplify the computation of the derivative, and does not affect the location of the extrema. As opposed to the condition number, which is dimensionless, \mathcal{G} is scaled by the translational speed, $\|\mathbf{T}\|$. We will therefore minimize it on the space of translation directions, that is, on the Gaussian sphere.
- The ratio $\text{Det } \mathbf{Q} / \text{Tr } \mathbf{Q}$ has the advantage of reaching its (theoretical) absolute maximum for $\lambda_1 = \lambda_m$ and its absolute minimum for λ_1 , just as the condition number does. It does not require a ranking of the eigenvalues; it can be computed directly without diagonalizing \mathbf{Q} ; finally, we can derive it to study its extrema, which is the problem we will address in the next section.

3.3 Deriving \mathcal{G}

We use the notation of [Roger80] with regard to matrix derivatives. Since $\|\mathbf{T}\| = 1$, we can represent \mathbf{T} as a function of the spherical angle vector $\mathbf{u} = (\theta, \phi)'$, with respect to which we perform the minimization

$$\mathbf{T}(\mathbf{u}) = (\cos \phi \sin \theta, \sin \phi \sin \theta, \cos \theta)'$$

The function to differentiate is

$$\mathcal{G} \circ \mathbf{Q}(\mathbf{u}) = \mathcal{G} \left(\sum_{l=1}^n \mathbf{S}_l \cdot \mathbf{T}(\mathbf{u}) \cdot \mathbf{T}'(\mathbf{u}) \cdot \mathbf{S}_l' \right).$$

Applying the chain rule for matrix differentials, we get

$$\frac{\partial \mathcal{G}(\mathbf{Q}(\mathbf{u}))}{\partial \mathbf{u}} = \left[\frac{\partial \mathcal{G}(\mathbf{X})}{\partial \mathbf{X}} \right]_{\mathbf{X}=\mathbf{Q}} * \left(\frac{\partial \mathbf{Q}(\mathbf{u})}{\partial \mathbf{u}} \right).$$

We chose \mathcal{G} in part for the simplicity of its derivatives:

$$\begin{aligned}\left[\frac{\partial \mathcal{G}(\mathbf{X})}{\partial \mathbf{X}} \right]_{\mathbf{X}=\mathbf{Q}} &= \left[[(\mathbf{X})^{-1}]' - \frac{1}{\text{Tr } \mathbf{X}} \cdot \mathbf{I}_m \right]_{\mathbf{X}=\mathbf{Q}} \\ &= [(\mathbf{Q}(\mathbf{u}))^{-1}]' - \frac{1}{\text{Tr } \mathbf{Q}(\mathbf{u})} \cdot \mathbf{I}_m.\end{aligned}$$

Inversion of \mathbf{Q} is justified since we expect to find nonzero eigenvalues. In order to simplify the computations, \mathbf{Q} can be expressed as a sum of matrices:

$$\mathbf{Q}(\mathbf{u}) = \sum_{l=1}^n \mathbf{Q}_l(\mathbf{u}) \quad \text{where} \quad \mathbf{Q}_l = \mathbf{S}_l \cdot \mathbf{T}_2 \cdot \mathbf{S}_l' \quad \text{and} \quad \mathbf{T}_2 = \mathbf{T} \cdot \mathbf{T}'.$$

$\partial \mathbf{Q} / \partial \mathbf{u}$ is computed as the sum of the $(\partial \mathbf{Q}_l / \partial \mathbf{u})_{l \in \mathbf{N}_n}$ and

$$\begin{aligned}\frac{\partial \mathbf{Q}_l(\mathbf{u})}{\partial \mathbf{u}} &= \frac{\partial}{\partial \mathbf{u}} (\mathbf{S}_l \cdot \mathbf{T}_2 \cdot \mathbf{S}_l') \\ &= (\mathbf{S}_l \otimes \mathbf{I}_2) \cdot \frac{\partial \mathbf{T}_2}{\partial \mathbf{u}} \cdot (\mathbf{S}_l' \otimes \mathbf{I}_1) \\ &= (\mathbf{S}_l \otimes \mathbf{I}_2) \cdot \frac{\partial \mathbf{T}_2}{\partial \mathbf{u}} \cdot \mathbf{S}_l'\end{aligned}$$

Finally, we obtain

$$\begin{aligned}\left\{ [(\mathbf{Q}(\mathbf{u}))^{-1}]' - \frac{1}{\text{Tr } \mathbf{Q}(\mathbf{u})} \cdot \mathbf{I}_m \right\} \\ * \sum_{l=1}^n ((\mathbf{S}_l \otimes \mathbf{I}_2) \cdot \mathbf{T}_{2\mathbf{u}} \cdot \mathbf{S}_l') = 0. \quad (10)\end{aligned}$$

which is the condition for the existence of an extremum (whether an extremum \mathbf{u}_0 is a maximum or a minimum is then settled by analysis of the eigenvalues of the Hessian matrix at \mathbf{u}_0).

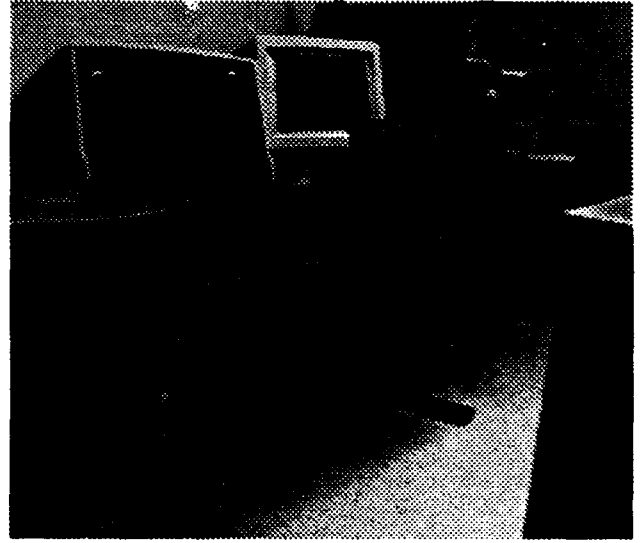


Figure 3 — One of the 640 × 480 images used for the experiments

3.4 Experimental results: Part I

In this series of experiments, the condition number of the \mathbf{Q} matrix was computed for a 640 × 480 image such as the one shown in Figure 3. The plotted surface corresponds to a scanning of the translation direction over the half Gaussian sphere, that is, $[-\pi/2, \pi/2] \times [-\pi/2, \pi/2]$.

In the case of the plot shown in Figure 4, the measuring functions μ_l were defined over 128 × 128 image windows, and were generated by all combinations of the form $\cos(ax + by + c)$, with a and b chosen among $\{0, 5, 20, 45, 100\}$, and $c \in \{0, -\pi/2\}$. The "reconstruction" functions ψ_k were defined to be the unit function over 32 × 32 image windows, thus defining a coarser grid over the image.

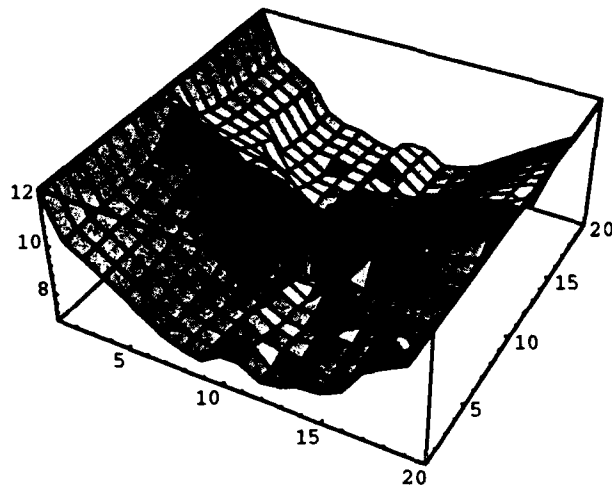


Figure 4 — Condition number for data set 1

In the case of the plot shown in Figure 5, a coarser 64×64 reconstruction grid was used. The measuring functions μ_i were still defined over 128×128 image windows, generated by all combinations of the form $\cos(ax + by + c)$. This time, however, a larger number of measuring functions was used, thus capturing more information about the image. Coefficients a and b were picked in $\{0, 3, 5, 11, 20, 45, 60, 100\}$, and $c \in \{0, -\pi/2\}$.

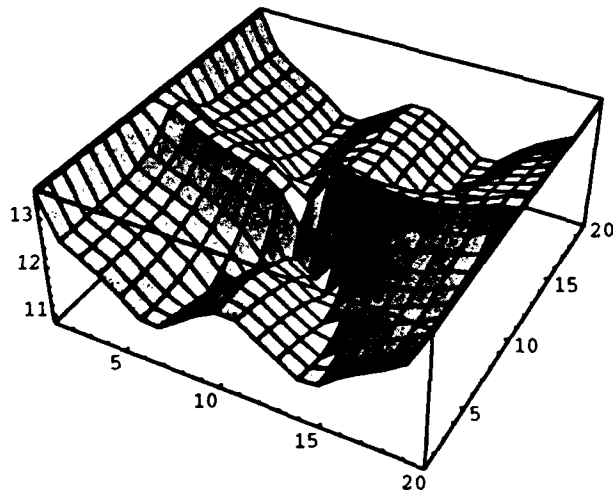


Figure 5 — Condition number for data set 2

One of the most important conclusions we can draw from the experiments we have performed so far is that the resolution of the reconstruction grid, as well as the number of measuring functions, if they affect the general shape of the plotted surface, do not seem to have any significant influence on the *locations* of strong minima. The validity of this conclusion is much easier to see when the problem is reduced to the case of a robot constrained to move in a plane, as we shall see in the next subsection.

3.5 Application to the case of a mobile robot

A simplified form of equation (10) is obtained in the case of a mobile robot since the translation displacements then take place in a plane. The translation vector can thus be described by a single heading angle: $T = \|T\| \cdot (\cos \phi, \sin \phi, 0)'$. Equation (10) then simplifies to the following scalar equation:

$$\text{Tr}(U \cdot V) = 0, \quad (11)$$

where

$$U = ((Q(\phi))^{-1})' - \frac{1}{\text{Tr} Q(\phi)} \cdot I_m$$

and

$$V = \sum_{l=1}^{l=m} (S_l \cdot T_{2\phi} \cdot S_l')$$

Let us consider a scene such as the one shown in Figure 6. The camera's optical axis is parallel to the motion surface, which means that the focus of expansion corresponding to the translation is situated on the x axis (the white horizontal line in the image).

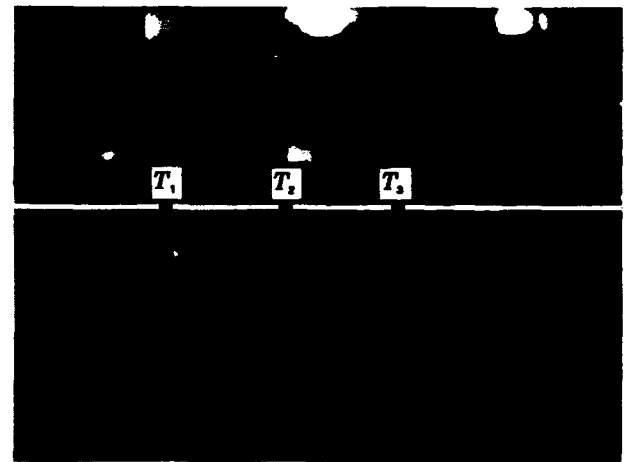


Figure 6 — Directions of translation for a mobile robot

What G provides is an estimate of which activities are *informative* for the robot, in the sense that they make its visual algorithms more robust, and hence, their results more reliable. It should therefore be used as an additional constraint by the planning algorithm, the preimage backchaining algorithms ([Latom89]). An even better use of G could be made by artificial potential field algorithms ([Khati86]): In addition to the attraction force exerted by the goal and the repulsion forces exerted by the obstacles, the robot could be made to be influenced by an "information" force, aiming at maximizing the quality of the data collected by the robot's visual sensors. This extension of the original planning algorithms would be feasible whether the visual data exploited by the controller is a classical depth map or the free space doors described in [Hervé91].

3.6 Experimental results: Part II

As was the case with general 3D motion (over the Gaussian sphere), linear features were defined by measuring functions μ_i of the form $\cos(ax + by)$ or $\sin(ax + by)$ (with a and b belonging to $\{0, 3, 5, 11, 20, 45, 60, 100\}$), whose domains were square windows over the image. The "reconstruction" functions ψ_k were again chosen to be unit functions defined over square image windows.

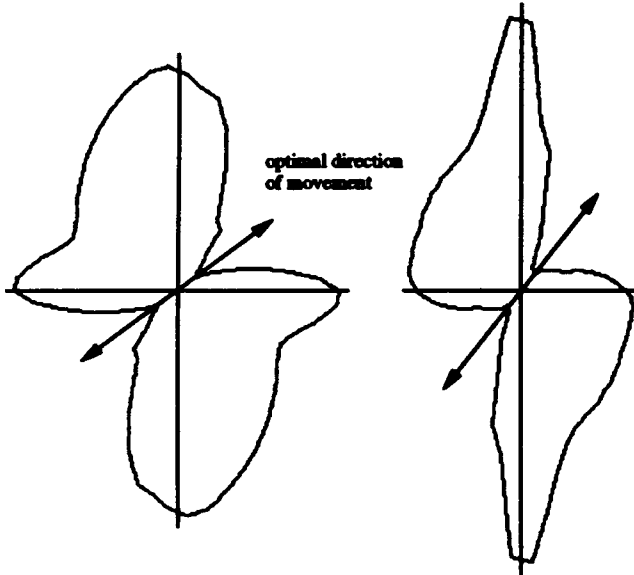


Figure 7 — (a) Plot for data set 1. (b) Plot for data set 2

Table 1 — Parameters for the first image

Fig.	μ_i windows	a and b in	ψ_k windows
7(a)	128×128	$\{0, 3, 5, 11, 20, 45, 60, 100\}$	16×16
7(b)	128×128	$\{0, 3, 5, 11, 20, 45, 60, 100\}$	64×64
8(a)	256×256	$\{0, 3, 5, 11, 20, 45, 60, 100\}$	64×64
8(b)	128×128	$\{0, 5, 20, 45, 100\}$	32×32
9(a)	64×64	$\{0, 5, 11, 20, 45, 60\}$	32×32
9(b)	128×128	$\{0, 5, 20, 33, 45, 100\}$	8×8

The first series of experiments were performed with the 640×480 image shown in Figure 3; their results are presented in Figures 7, 8, and 9. Table 1 gives, for each figure, the size of the linear feature window, the parameters used to generate the measuring functions, and the size of the reconstruction window.

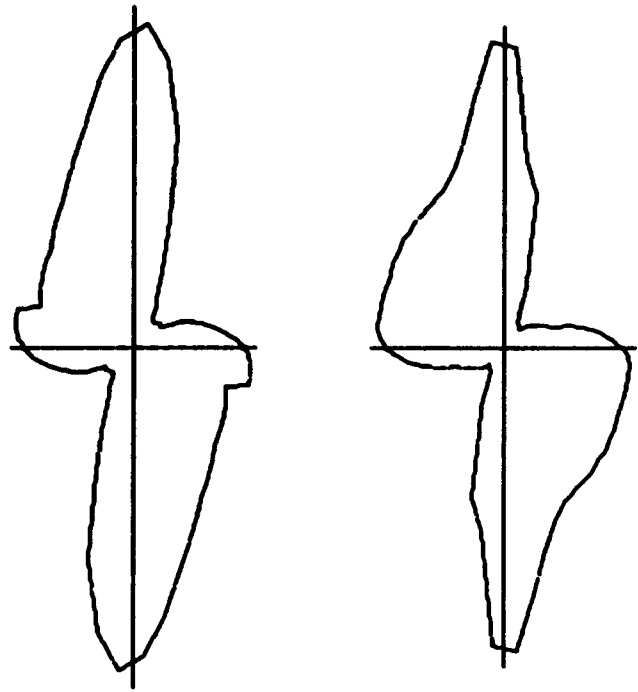


Figure 8 — (a) Plot for data set 3. (b) Plot for data set 4

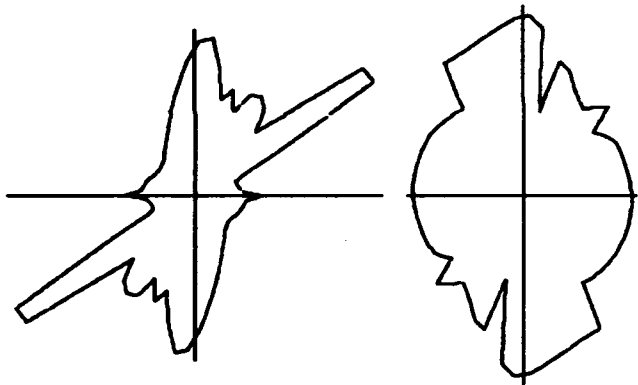


Figure 9 — (a) Plot for data set 5. (b) Plot for data set 6

The first conclusion we can draw from these six parametric (polar) curves derived from our experiments is that, for given choices of linear feature vector and of reconstruction function modelling the depth map, there are directions of motion—actions—for which the visual algorithms perform much better than for others. Since the curves plot the condition number of matrix Q , a point close to the origin corresponds to a more stable solution of the reconstruction equations; a displacement in that direction therefore corresponds to an optimization of the algorithmic stability of the visual module. Conversely, the points most distant from the origin correspond to "bad" actions of the observer, resulting in maximum algorithmic instability of the visual module, and should therefore be avoided.

The second conclusion suggested by our results is that the location of the best and worst directions of displacement (the strong extrema of the condition number) are not perturbed much by changes in the parameters of the visual algorithms:

- The resolution of the reconstruction grid, from 16×16 for Fig. 7(a) to 64×64 for Fig. 7(b). If these initial results are confirmed by further experiments, it would mean that the information we obtain for low resolutions about best and worst actions could extend to higher resolutions.
- As Figure 9(b) clearly shows, increasing the resolution of the reconstruction grid improves the details of the condition number curve, but at the cost of narrowing the "good" regions, now surrounded by sharp peaks of bad performance. This may not be a very surprising observation, judging by the difficulty encountered in all shape from x problems that aim at extracting dense depth maps.
- Similarly, changes in the linear feature vector have little effect on the location of strong extrema, whether the changes affect the size of the linear features' windows (compare Figures 7(b) and 8(a)) or the measuring functions themselves (Figures 7(a) and 8(b)).
- As with the resolution of the reconstruction grid, it is possible to change the linear feature vector so as to obtain a more detailed condition number curve, but again at the cost of increasing the overall instability and narrowing the "good" areas of the curve.

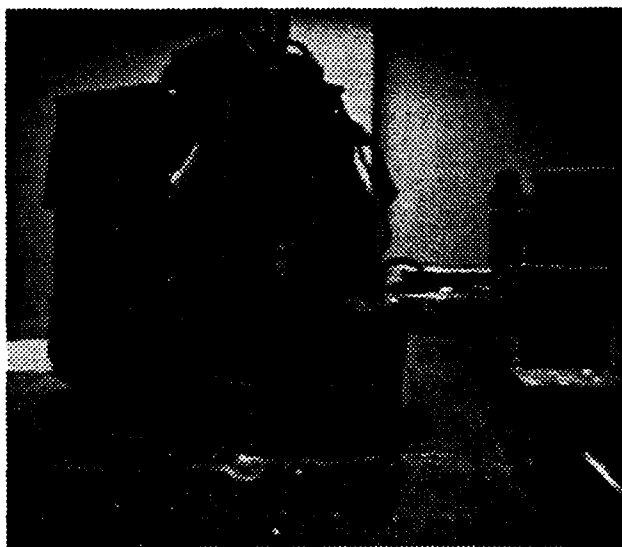


Figure 10 — Another example of 640×480 input image

Similar results have been obtained with other images, such as the 640×480 image in Figure 10. Table 2 gives for Figures 12, 11(a), and 11(b) the size of the linear feature window, the parameters used to generate the measuring functions, and the size of the reconstruction windows.

Table 2 — Parameters for the second image

Fig.	μ_i windows	a and b in	ψ_k windows
12	128×128	$\{0, 3, 5, 11, 20, 45, 60, 100\}$	64×64
11(a)	256×256	$\{0, 3, 5, 11, 20, 45, 60, 100\}$	64×64
11(b)	128×128	$\{0, 5, 20, 45, 100\}$	32×32

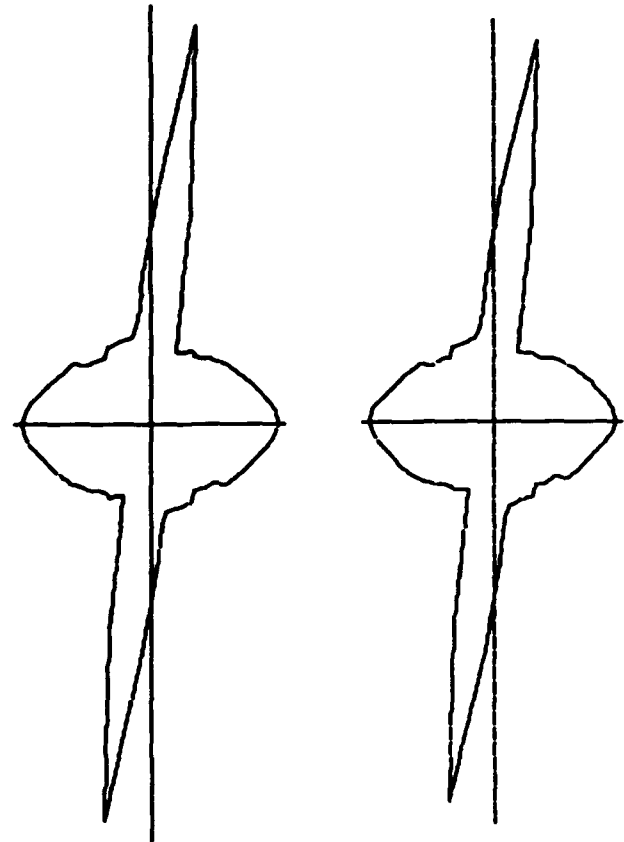


Figure 11 — (a) Plot for data set 1. (b) Plot for data set 2

4. Discussion

4.1 Motion and perception

The first part of this paper presented an active approach to the problem of visual reconstruction, which provides us with a *model-based* depth map of the observed scene. However, there are reasons to believe that a depth map may not be necessary to accomplish complex navigation tasks such as obstacle avoidance or visual servoing. In fact, the process in which the visual reconstruction community has been engaged over the last decade may prove much more worthwhile than the goal it

has set for itself. On the way to the realization of shape from x modules, we have learned about the computational issues of the visual process and have accumulated results and algorithms which all describe parts of the vision problem, and can be directly exploited in the control of robotic tasks.

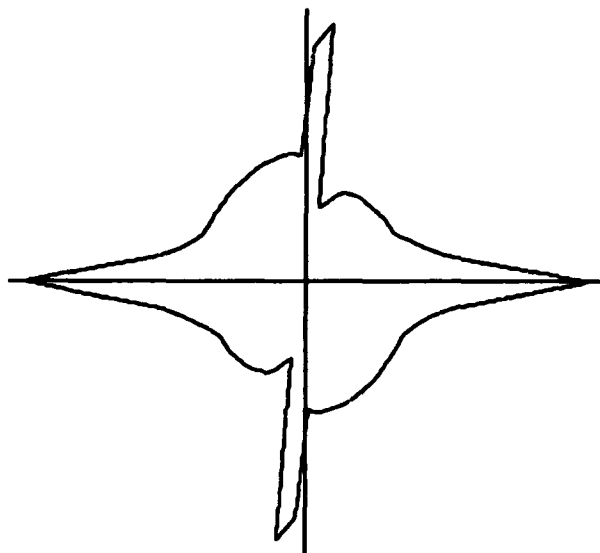


Figure 12 — Plot for data set 3

Traditionally, it has been assumed that the role of vision was to provide a depth map to a planning system which would then decide on a next move for the robot. This scheme implicitly incorporates elements of feedback (if only by the effects of the motion on the visual input), but it separates task planning from action and sensing too sharply. Clearly, the boundary between them is much fuzzier, and we would like sensing to take a greater part in the decision process.

Even if we avoid reconstructing a depth map, the algorithms we will be using in our systems will be based on reconstruction algorithms, since in one form or another, some three-dimensional information is needed, whether it is the *time to collision* or the distance to a particular feature of the workspace or of the configuration space. In this context, it becomes useful—and even necessary—to be able to determine *a priori* which action will result in good behaviors of the vision algorithms and which will provoke instability in the computations.

The determination of an “optimal” motion which we proposed in the previous section is a first step in this direction. Naturally, we cannot expect the planning module to apply this action, regardless of the task it has to accomplish, but it should treat as one of its *constraints* the need to keep the direction of displacement in the neighborhood of one of the “good” directions determined for the visual process.

4.2 Limitations and drawbacks of the method

As is always the case when one attempts to extract information from of real images, *ad hoc* (or if one prefers a gentler term, heuristic) choices had to be made in the work we presented here:

- We have already discussed the validity and limitations of the optical flow equation, whether used pointwise or integrated over image windows. The method we presented is not restricted to the use of the equation proposed in [HornS81]. Any more sophisticated constraint, however, involves the inconvenience of requiring additional (unknown and variable) parameters describing, for example, the reflective properties of the objects in the scene.
- As is generally the case when least squares optimization is used, the main justification for the choice of this criterion is that it keeps the mathematical expressions simple. We have at this point no way of estimating the independence of the variables
- Choosing the ψ_k functions to represent the depth poses more practical than theoretical problems, due to the fact that the number of such functions determines the size of the system of linear equations to solve. For example, if not for the prohibitive computational cost of such a choice, one could imagine defining the ψ_k 's to be constant functions (giving the value 1.0), over pixel-size windows.
- Choosing the measuring functions $(\mu_l)_{l=1,n}$, however, poses more serious theoretical problems: what types of functions capture the most information about the image, or about a class of images? Could this be determined *a priori*? Here again, the goal is to make matrix Q as well conditioned as possible, but one would now be looking for a minimax solution: the one giving the lowest condition number in the case of the worst observer motion (the “best” displacement may not be feasible in the context of the observer's task).

5. Conclusion

We have presented a theory about the extraction of the shapes of observed objects. Our approach combines the modules of shape from shading by fusing the information relevant to each of them, without having to resort to segmentation or explicit selection of one algorithm or another for a given area of the image. By using linear features (i.e. by considering variations of the intensity over areas and not at isolated points), we avoid the pointwise computation of unreliable operators, which is a flaw of classical methods. Finally, we show that the observer can determine a motion that optimizes the stability of the equation to be solved, and therefore the reliability of the solution.

APPENDIX A Derivation of Equation (5)

We are trying to prove that

$$\frac{d}{dt} \left[\iint_{\Sigma_i} I \mu_i ds \right] = \iint_{\Sigma_i} (I \nabla_j \mu_i \cdot \dot{\mathbf{m}}) ds$$

In order to do this, we have to express the conservation of a function \mathcal{A} over a 3D domain \mathcal{V} .

Conservation Law: Let \mathcal{V} be a compact subset of \mathbb{R}^3 with oriented boundary $\partial\mathcal{V}_+$, subject to deformations and displacements, and let \mathcal{A} be a function defined on \mathcal{V} . Then the following applies:

$$\frac{d}{dt} \left[\iiint_{\mathcal{V}} \mathcal{A} d\tau \right] = \iiint_{\mathcal{V}} \frac{\partial \mathcal{A}}{\partial t} d\tau + \iint_{\partial\mathcal{V}_+} \mathcal{A} \mathbf{V} \cdot d\mathbf{s}, \quad (\text{A.1})$$

where \mathbf{V} is the velocity at the boundary point under consideration and $d\mathbf{s}$ is directed along the outer normal.

N.B. The 2D equivalent of this law (i.e. where triple and double integrals are replaced by double and simple integrals respectively) is valid when the surface is planar and all deformations/motions occur in this plane. It is also valid for some pathological forms of \mathcal{A} , one of which, as will be shown here, happens to be our example ($\mu_i I$).

Ostrogradsky's divergence theorem tells that

$$\iint_{\partial\mathcal{V}_+} \mathbf{a} \cdot d\mathbf{s} = \iiint_{\mathcal{V}} \nabla_j \cdot \mathbf{a} d\tau.$$

Applying this to (A.1) yields:

$$\begin{aligned} \frac{d}{dt} \left[\iiint_{\mathcal{V}} \mathcal{A} d\tau \right] &= \iiint_{\mathcal{V}} \frac{\partial \mathcal{A}}{\partial t} d\tau + \iint_{\partial\mathcal{V}_+} \mathcal{A} \mathbf{V} \cdot d\mathbf{s} \\ &= \iiint_{\mathcal{V}} \left[\frac{\partial \mathcal{A}}{\partial t} + \nabla_j \mathcal{A} \cdot \mathbf{V} + \mathcal{A} \nabla_j \cdot \mathbf{V} \right] d\tau \\ &= \iiint_{\mathcal{V}} \left[\frac{d\mathcal{A}}{dt} + \mathcal{A} \nabla_j \cdot \mathbf{V} \right] d\tau. \quad (\text{A.2}) \end{aligned}$$

We now try to pick a \mathcal{V} such that this volume equation can be transformed into a surface equation for the particular type of problem we are studying here. In order to simplify the definition, we will call $\mathcal{C}(P, \mathcal{S})$ the cone of vertex P generated by the surface \mathcal{S} .

Σ_i is the window on which the linear feature is computed. We choose \mathcal{V} to be the part of $\mathcal{C}(O, \Sigma_i)$ lying between Σ_i and a surface Σ'_i defined as follows: for every surface element $ds \subset \Sigma_i$, the part of $\mathcal{C}(O, ds)$ included in \mathcal{V} has a volume equal to $1 \cdot ds$ (or $L \cdot ds$ for any given L , as long as Σ'_i lies between O and Σ_i).

If \mathcal{A} is such that $\mathcal{A}(X, Y, Z) = \mathcal{A}'(\frac{X}{Z}, \frac{Y}{Z})$ then (A.2) can be written as follows:

$$\frac{d}{dt} \left[\iint_{\Sigma_i} \mathcal{A}' ds \right] = \iint_{\Sigma_i} \left[\frac{d\mathcal{A}'}{dt} + \mathcal{A}' \nabla_j \cdot \mathbf{V} \right] ds.$$

If the displacement of \mathcal{V} is rigid ($\mathbf{V} = \mathbf{T} + \boldsymbol{\omega} \times \mathbf{M}$) then $\nabla_j \cdot \mathbf{V} = 0$ and the conservation law (A.1) simplifies to

$$\frac{d}{dt} \left[\iint_{\Sigma_i} \mathcal{A}' ds \right] = \iint_{\Sigma_i} \frac{d\mathcal{A}'}{dt} ds.$$

In the case that interests us, $\mathcal{A}'(x, y) = I(x, y) \cdot \mu_i(x, y)$ (the light intensity along a ray passing through the optical center is considered constant) and $\varphi_i = \iint \mathcal{A}' ds$.

The expression turns out to be remarkably simple:

$$\begin{aligned} \frac{d\mathcal{A}'}{dt} &= \frac{dI}{dt} \mu_i + I \nabla_j \mu_i \cdot \dot{\mathbf{m}} \\ &= I \nabla_j \mu_i \cdot \dot{\mathbf{m}}. \end{aligned}$$

which, combined with (A.2), gives the expected result.

References

- [Aloim90] Y. Aloimonos, "Purposive and qualitative active vision," in *Proceedings DARPA Image Understanding Workshop*, pp. 816-828, Pittsburgh, Pennsylvania, 1990.
- [Aloim89] Y. Aloimonos, "Unifying shading and texture through an active observer," *Proceedings of the Royal Society of London, B* 238: 25-37, 1989.
- [Aloim88a] Y. Aloimonos and A. Basu, "Combining information in low-level vision," in *Proceedings DARPA Image Understanding Workshop*, pp. 862-906, Cambridge, Massachusetts, 1988.
- [Aloim88b] Y. Aloimonos, I. Weiss, and A. Bandopadhyay, "Active vision," *International Journal of Computer Vision*, 2: 333-356, 1988.
- [Amari86] S. Amari, personal communication, 1986.
- [Bajcs86] R. Bajcsy, "Passive perception vs. active perception," in *Proceedings IEEE Workshop on Computer Vision*, Ann Arbor, Michigan, 1986.
- [Chati85] R. Chatilla and J.P. Laumond, "Position referencing and consistent world modeling for mobile robots," in *Proceedings 1985 IEEE International Conference on Robotics and Automation*, pp. 138-145, St. Louis, Missouri, 1985.
- [Fedde89] J.T. Feddema and O.R. Mitchell, "Vision-guided servoing with feature-based trajectory generation," *IEEE Transactions on Robotics and Automation*, 2: 691-700, 1989.
- [Gibso50] J.J. Gibson, *The Perception of the Visual World*, Riverside Press, Cambridge, Massachusetts, 1950.
- [Hervé91] J-Y. Hervé, "Visual feedback for autonomous navigation," in *Proceedings IEEE International Conference on Systems, Man, and Cybernetics*, pp. 219-224, Charlottesville, Virginia, 1991.
- [HornS81] B.K.P. Horn and B. Schunck, "Determining optical flow," *Artificial Intelligence*, 2: 185-204, 1981.
- [Horn77] B.K.P. Horn, "Image intensity understanding," *Artificial Intelligence*, 2: 201-231, 1977.
- [Khati86] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, 2: 90-99, 1986.

- [Latom89] J-C. Latombe, "Motion planning with uncertainty: on the preimage backchaining approach," in *The Robotics Review 1*, O. Khatib, J.J. Craig, and T. Lozano-Pérez, editors, M.I.T. Press, Cambridge, Massachusetts, 1989.
- [Marr82] D. Marr, *Vision*, Freeman, San Francisco, California, 1982.
- [Poggi85] T. Poggio, V. Torre and C. Koch, "Computational vision and regularization theory," *Nature*, 2: 314-319, 1985.
- [Roger80] G.S. Rogers, *Matrix Derivatives*, Marcel Dekker, New York, 1980.
- [Sande83] A.C. Sanderson and L.E. Weiss, "Adaptive visual servo control of robots," in *Robot Vision*, A. Pugh, editor, Springer-Verlag, New York, 1983.
- [Shulm89] D. Shulman and J-Y. Hervé, "Regularization of discontinuous flow fields," in *Proceedings IEEE Workshop on Visual Motion*, pp. 81-86, Irvine, California, 1989 .
- [Shulm88] D. Shulman and Y. Aloimonos, "Boundary preserving regularization: Theory, Part I," CAR-TR-356, Center for Automation Research, University of Maryland, College Park, Maryland, April 1988.
- [TsaiL89] R.Y. Tsai and R.K. Lenz, "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration," *IEEE Transactions on Robotics and Automation*, 2: 345-358, 1989.
- [TsaiH84] R.Y. Tsai and T.S. Huang, "Uniqueness and estimation of three dimensional motion parameters of rigid objects with curved surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2: 13-27, 1984.
- [Weiss84] L.E. Weiss, "Dynamic Visual Servo Control of Robots: An Adaptive Image-based Approach," Ph.D. Thesis, Dept. of ECE, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.

Planning and Selective Perception for Target Retrieval

Theodore Camus Jonathan Monsarrat Thomas Dean*

Department of Computer Science
Brown University, Box 1910, Providence, RI 02912

Abstract

Target retrieval tasks characterize an important class of problems in mobile robotics. In such problems, a robot searches through a cluttered environment and identifies objects matching a specified description. To speed search, the robot should avoid using slow, high-accuracy sensing in regions that are unlikely to contain the target. We describe an approach in which search focuses on probable regions found with inexpensive sensing. Decision making is performed by three modules: a high-level decision making component based on Bayesian decision theory provides the overall search strategy, a path planner adds navigational refinements, and a low-level controller executes the strategy while coping with obstacles and unexpected events. We describe an implementation of our approach and a series of initial experiments.

1 Introduction

Many robotics applications require a mobile robot to fetch an instance of a particular type of object in an unfamiliar environment. Such *target retrieval* tasks represent an important class of problems in mobile robotics. For example, researchers in a lab may ask a robot assistant to locate and retrieve a particular piece of equipment while they attend to more important matters. Similarly, workers at a construction site

might use robots to fetch tools and building materials. To expedite search the robot must carefully deploy its sensors to deal with the uncertainty in sensing.

In the target retrieval tasks considered in this paper, we assume that the boundary of the search area is known but that there is little if any prior information concerning the interior of the search area. In particular, there is no prior information concerning the location of the target or locations of other objects (possibly similar in appearance to the target) that may impede exploration or occlude the target. Due to the computational cost of high-accuracy object recognition routines, we wish to use such routines sparingly; the robot searches those areas that are believed with high probability to contain the target, as determined by lower-cost, less accurate sensing routines.

Our robot must also deal with the problem of navigation. In particular, it must move from one location to another efficiently while avoiding obstacles along the way. This involves both high-level path planning and low-level navigation routines, which must be able to communicate in a straightforward manner. The path planner itself is required to build and maintain a map that facilitates both planning and navigation.

In the next section, we describe a three-level solution to the task retrieval problem. Then we consider the decisions made by each level. Finally, we describe our experiments in active perception and present our conclusions.

*This work was supported in part by a National Science Foundation Presidential Young Investigator Award IRI-8957601, by the Advanced Research Projects Agency of the Department of Defense monitored by the Air Force under Contract No. F30602-91-C-0041, and by the National Science foundation in conjunction with the Advanced Research Projects Agency of the Department of Defense under Contract No. IRI-8905436.

2 Proposed Solution

Our solution consists of three levels of control: *supervisory control*, *path planning*, and *low-level control*. Each level has its own spatial representation that is consistent with the other two levels yet is appropriate for its particular scale of spatial reasoning.

The highest level of representation is the *supervisory control*. The supervisor directs the robot to square areas called *regions* believed with high probability to contain the target object. This belief is based on sensor information provided by the robot itself, and may include data acquired in the course of the robot's *low-level control* phase.

To reduce computational complexity, these regions should be large. In our case the practical limit of our sensors' ability to detect objects is five meters, so these regions are modeled as square areas approximately four meters on a side. These areas can be efficiently represented by a coarse *occupancy grid* [Moravec and Elfes, 1985], with each grid element corresponding to a region. The *path planning* module has some freedom to direct the search because it is not told exactly where in the region to go. The expected time to find the target is reduced by first choosing regions likely to contain it.

The path planner directs the robot's route to the region given by the supervisory control. A direct path may be impossible, risky, or inefficient in certain circumstances. Path planning could use a direct model of each obstacle's boundary, but this much detail is not required. It is much more convenient to model important locations in the world with *nodes*. Pathways between locations are represented by edges in a graph.

To use this model for path planning, we make the assumption that traversing an edge requires only local information. This allows us two advantages. First, we are able to separate the low-level control module from the rest of the system. Thus, multiple low-level control routines may be selected for different circumstances. Second, we

have abstracted away detail which would make path planning expensive. The low-level control module need only give the path planner the estimated time cost of traversing the given pathway.

The *low-level control* traverses edges efficiently, avoiding any obstacles in the way. Low-level control requires its own spatial model. Our sensors return obstacle information as the common data structure that connects low-level control with the path planner. Our robot uses a laser light striper for ranging information, along with near infrared sensors for obstacle avoidance and navigation information.

Once the robot has arrived at a location, search for the target object can proceed. A list of candidate target objects is generated using a quick scan with the light striper. The robot can then move adjacent to the object and examine it with its stereo camera pair. A stereo algorithm is used to recover the target's 3D structure. A set of algebraic invariants is then computed by comparing the recovered structure with that associated with the object to be recognized, which is stored in a database. If the computed invariants match the ideal ones, the target is found and a successful recognition is signaled [Subrahmonia *et al.*, 1992]. Otherwise, this step is repeated for each object in the candidate list until the target is found. If the target is not found in the candidate list, the robot continues the search in a new area as directed by the supervisor.

Once the robot has arrived at the region given by the supervisory control, search for the target object can proceed. A list of candidate target objects is generated using a quick scan of the robot's sensors. The robot can then move adjacent to the object and examine it with its stereo camera pair.

2.1 Supervisory Control

It is the responsibility of the supervisor to generate strategies for finding the target as soon as possible. To expedite search, the supervisor combines fast but error-prone sensing routines with slower but more accurate sensing routines. Each strategy corresponds to a sequence

of navigation and sensing actions. The supervisor uses a *temporal Bayesian network* [Dean and Kanazawa, 1989] to compute the value of various information gathering strategies. From a given initial situation, the robot selects a set of reasonable action sequences from a library of such sequences. The expected value of an action sequence is measured in terms of the amount of information it provides about the location of the target.

From the previous history of fast inaccurate sensing, the supervisor has a distribution of places to visit and costs to arrive at those places. It chooses the visitation sequence with the highest expected value, and lets the path planner execute the first visit. Subsequent actions in the chosen sequence may not occur. The lower levels of the system execute and report on the success of the first visit. Success provides a list of objects observed in the regional visit, with a measure of their similarity to the target. This list is added to the supervisory model, and used to update the beliefs about the target's location. The supervisor then creates a new visitation sequence if needed.

2.2 Path Planning

A path consists of a sequences of *nodes* connected by *edges*, which may be *real*, if they have been traversed and are known to exist, or *virtual*, if they are unknown quantities. The *cost* of a path is some non-global function of its component edges. Each edge has a *weight* associated with it, determined by the underlying edge data, and used by the path planner. For example virtual edges have greater weight than real edges. Edges that cannot be traversed due to obstacles are considered to have an infinite weight.

The path planner is required to quickly return a path with low cost. We use a variant of best-first search and a set of heuristics to eliminate paths that are unnecessarily complicated. When the supervisor directs the planner to move to a given region, the planner selects a destination node corresponding to a location in that region if one already exists or creates a new one otherwise.

Path traversal consists of traversing each of the component edges of the path. At each node, the robot must realign itself toward the next node and proceed if there are no obstacles in the way. The path planner itself has insured to the best of its knowledge that there are no obstacles in any of the component edges. can consult the occupancy grid for an estimate of the there being an obstacle in any given untraversed planner assumes that the low-level control system is able to deal with obstacles encountered in traveling from one node to another. Additional details regarding the path planner are in the longer version of this paper [Camus *et al.*, 1993].

2.3 Low-level Control

The purpose of the low-level control system is to execute the path selected by the path planner. Execution consists of visiting in sequence each component node of the chosen path, avoiding all obstacles along the way. Each pair of nodes in the path are connected by an *edge* representing a pathway that contains all the information necessary to get from one node to another.

The low-level control system relies on deadreckoning and the accuracy of a laser-light-striper ranging system to reliably traverse edges in the network of nodes. Accurate edge traversal is essential to maintain registration with the map (network of nodes) being constructed during the search for the target.

3 Experiments

In this section we describe a series of experiments involving, Gort, a mobile robot especially designed for this research. Gort is operates in an enclosed atrium filled with boxes and oddly shaped obstacles. A target shaped like the obstacles is located in the area, and Gort searches the area to locate the target. Gort uses stereo vision both to identify candidate target objects and to discriminate between the targets and obstacles.

3.1 Hardware

Gort consists of a circular, four-wheeled base, an on-board computer, four sensors, and a

power supply. The base, from Real World Interface, includes its own controller and communicates with the on-board computer over a standard serial line. The base carries large batteries that can power the robot for up to five hours. Resting on the base is a metal cage that supports the additional hardware of the robot.

The on-board computing equipment consists of a VME card cage containing a 68030 processor board with a hard drive. The processor board runs a UNIX variant that we can port software to directly from our workstation development platforms. All the navigation and planning algorithms run on the 68030 processor. However, a tether connects the robot to a separate workstation for handling the vision routines.

Two sensors are memory-mapped into the virtual memory of the on-board computer: a ring of infrared sensors and a laser light stripper. The infrared sensors are good proximity sensors, providing more robust sensing of obstacles than most standard acoustic sensors. The laser light stripper is a good medium-range sensor for detecting and measuring the shape of obstacles.

Also mounted on the metal cage are two cameras, angled for stereo vision. The cameras are connected directly to high-bandwidth cables that feed the vision information directly to a remote computer for processing. The cables also provide communication between the remote computer and the on-board computer.

3.2 Target Recognition

A key component to the system is the acquisition of probabilistic information for the temporal Bayesian network planner. The ability of the supervisor to deal with noisy data is essential in the overall success of the system. We use two sensor strategies to deal with this problem. First, the laser light stripper is used to detect objects within the robot's same region. The laser can detect objects with high accuracy within this range. A full sweep with the laser thus constitutes a thorough search of the robot's own region.

Unfortunately the laser light stripper's effective

range is limited, so to detect potential target's at long range we use vision. We use a pattern-matching algorithm which seeks to identify potential targets at up to 8 meters away. This data is not as complete as that from the laser light stripper, and serves to inform the supervisory control of potential targets in neighboring regions. The supervisor considers the information from both these sensors in deciding where to direct the robot for future search.

3.3 Results

Our primary results concern the navigation components: the path planning and low-level controls. Each has been successfully tested in both laboratory and target environments.

The path planner has successfully dealt with sensing uncertainty and real time demands for robot control. Simulated results have shown the path planner to be a reliable method of returning efficient paths even when dealing with thousands of locations. Physical results have shown the ability of the path planner and the low-level control to coordinate efficiently through a field of obstacles.

The low-level obstacle avoidance algorithms have been successful in all experiments. Two methods of using the light stripper to detect obstacles were originally developed. One method makes use of the laser's wide-angle beam to detect objects in the periphery. Unfortunately the laser stripe is thin and less visible off-center, making additional checks necessary for accurate sensing. These checks can increase the time it takes to detect obstacles.

Thus, we have chosen the second method, which is to rotate the robot base and sweep the center of the laser beam (where it is strongest) back and forth across the search area. This approach was very successful in detecting the boundaries of flat surfaces and boxes, and even high-curvature objects such as tree planters. Unfortunately, turning the robot to sense better led to rotational inaccuracy in the robot dead reckoning.

The integration of the navigational modules has

been seamless. Once created, a node provides a convenient way around an existing obstacle. Future trips can make use of the node in the path planning stage so that the obstacle avoidance stage can be skipped for these previously traversed paths.

Preliminary experiments concerning the data acquisition for the temporal Bayesian network Supervisor have been successful in both laboratory and target environments. The Bayesian network itself has also been successfully simulated.

4 Related Work

Dean and Wellman [1991] describe our basic approach to planning using Bayesian networks. Levitt *et al.* [1988] discuss issues involving search in the context of object recognition. Rimey [1992] consider the problems involved in repositioning a robot head for recognition purposes. Agosta [1991] discusses some of the more subtle issues involved in quantifying relations among visual features.

Basye *et al.* [1992] provide an overview of how the Bayesian approach can be applied to problems in robotics with additional details in [Dean *et al.*, 1990]. Kirman *et al.* [1991] discuss the use of sensor abstractions to deal with the combinatorics of using Bayesian networks.

References

- [Agosta, 1991] Agosta, J. M. 1991. Conditional inter-causally independent' node distributions, a property of 'noisy-OR' models. In *Proceedings of the Seventh Conference on Uncertainty in AI*. 9-16.
- [Basye *et al.*, 1992] Basye, Kenneth; Dean, Thomas; Kirman, Jak; and Lejter, Moises 1992. A decision-theoretic approach to planning, perception, and control. *IEEE Expert* 7(4):58-65.
- [Camus *et al.*, 1993] Camus, Theodore; Monsarrat, Jonathan; and Dean, Thomas 1993. Planning and Selective Perception for Target Retrieval. Brown University Technical Report.
- [Dean and Kanazawa, 1989] Dean, Thomas and Kanazawa, Keiji 1989. A model for reasoning about persistence and causation. *Computational Intelligence* 5(3):142-150.
- [Dean and Wellman, 1991] Dean, Thomas and Wellman, Michael 1991. *Planning and Control*. Morgan Kaufmann, San Mateo, California.
- [Dean *et al.*, 1990] Dean, Thomas; Camus, Theodore; and Kirman, Jak 1990. Sequential decision making for active perception. In *Proceedings of the DARPA Image Understanding Workshop*. DARPA. 889-894.
- [Kirman *et al.*, 1991] Kirman, Jak; Basye, Kenneth; and Dean, Thomas 1991. Sensor abstractions for control of navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*. 2812-2817.
- [Levitt *et al.*, 1988] Levitt, Tod; Binford, Thomas; Ettinger, Gil; and Gelband, Patrice 1988. Utility-based control for computer vision. In *Proceedings of the 1988 Workshop on Uncertainty in Artificial Intelligence*.
- [Moravec and Elfes, 1985] Moravec, H. P. and Elfes, A. 1985. High resolution maps from wide angle sonar. In *IEEE International Conference on Robotics and Automation*. 138-145.
- [Rimey, 1992] Rimey, Raymond D. 1992. Where to look next using a bayes net: An overview. In *Proceedings of the DARPA Image Understanding Workshop*. DARPA. 927-932.
- [Subrahmonia *et al.*, 1992] Subrahmonia, J.; Cooper, D. B.; and Keren, D. 1992. Practical reliable bayesian recognition of 2-d and 3-d objects using implicit polynomials and algebraic invariants. Technical report lems-107, Brown University. Under review for publication in the *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Dynamic Sensor Planning

Steven Abrams Peter K. Allen
Center for Research in Intelligent Systems
Computer Science Department
Columbia University
New York, NY 10027

Konstantinos A. Tarabanis
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

Abstract

In this paper, we describe a method of incorporating the sensor planning abilities of the "MVP" Machine Vision Planning system to function in a dynamic robot work-cell. By mounting a camera on a manipulator, it is possible to compute a series of viewpoints and to move the camera to them at appropriate times so that there is always a robust view suitable for monitoring a robot task. The dynamic sensor planning system presented here achieves this by analyzing geometric models of the environment and of the planned motions of the robot, as well as optical models of the camera itself. It computes a series of viewpoints, each of which provides a valid viewpoint for a different interval of the planned task. Experimental results monitoring a robot operation are presented, and directions for future research are discussed.

1 Introduction

Recently, there has been much research in the field of sensor planning [Cowan and Bergman, 1989, Hutchinson and Kak, 1989, Ikeuchi and Kanade, 1989, Tarabanis *et al.*, 1991a]. The basic problem is that in setting up an automated system for monitoring some process, the effectiveness of the system can largely be determined by the locations, types and configurations of the sensors used. To manually determine these parameters on a case by case basis may not be cost effective or accurate, and the resulting system may not be optimal in any sense. It may be better to have an automated system for determining the sensor locations and parameters for monitoring a given task.

To that end, many systems have been and are

being developed which, based on geometric models of an environment and models of the sensors, can generate sensor locations and settings which provide a robust view of specific features so that the features are detectable, recognizable, measurable, or meet some other task constraints. In general, the sensors are cameras and a robust view implies that the camera must have an unobstructed view of the entire feature set, which must lie within the depth-of-field of the camera and must be magnified to a given specification. Sensor planning systems can then generate camera locations, orientations, lens settings (focusing adjustment, focal length, aperture), and in some cases lighting plans to insure a robust view of the features.

It is interesting to note that while research in robot motion planning abounds, research in sensor planning has focused on sensor planning for static scenes. It is our belief that an intelligent robot system capable of planning its own actions should be capable of planning its own sensing strategies. With a dynamic sensor planning system, this goal is closer to a reality. Robots involved in manufacturing or assembly can determine appropriate sensor locations. Teleoperators can have the robot system guarantee robust viewpoints during the operation. The intelligent motion plans which researchers spend so much effort computing can be monitored in an intelligent fashion.

To that end, we have been exploring methods of extending the sensor planning abilities of the "MVP" Machine Vision Planning [Tarabanis, 1991, Tarabanis *et al.*, 1991a] system to function in environments where objects are moving. In particular, we focus on sensor planning in a

dynamic robotic work cell environment.

In previous work, we described a technique for sensor planning in a dynamic environment [Abrams and Allen, 1991], which was implemented using a simulated model of a simple moving object. Here, we present a detailed analysis of the dynamic sensor planning problem and improved versions of the original algorithms. In addition, experimental results using a model of a dual-robot work cell are presented in which we automatically monitor a task in the work cell.

2 Overview of Static Planning

A complete description of the MVP system is beyond the scope of this paper. For details, see [Tarabanis, 1991, Tarabanis *et al.*, 1991a, Tarabanis *et al.*, 1991b]. In brief, MVP takes a constraint based description of the vision task requirements and synthesizes what has been termed a *generalized viewpoint*, which is an eight-dimensional vector incorporating sensor location, orientation, and lens parameters including aperture and effective focal length. The constraints MVP considered in determining viewpoints are depth-of-field, field-of-view, resolution, and unoccluded visibility.

MVP contains analytical relationships for the optical task constraints (resolution, focus, field-of-view), and uses 3-D solid geometric models of the environment to formulate visibility constraints. (The geometric models are polyhedra, both convex and concave.) The constraint equations can be thought of as defining hypersurfaces bounding feasible regions in the 8-dimensional parameter space of the generalized viewpoint. These constraints are combined in an optimization setting to produce a generalized viewpoint which meets all task constraints with as much margin for error in sensor placement and setting as possible (i.e., as far away from all hypersurfaces as possible). Using CAD descriptions of the object to be viewed and its environment, MVP generates the visibility region for viewing the desired features. This region is calculated to be the total volume in space from which the features are viewable without obstruction. This volume is used in the optimization stage of MVP for finding the best viewpoint.¹

¹ Here, and elsewhere in this paper, when we refer to a view-

3 Motion in the Work Cell

There are two basic cases which must be dealt with separately in the dynamic sensor planning problem. First is the case where the target objects, i.e. those features which must be viewed, remain stationary and other objects, such as the robot which is performing some operation on the stationary part, moves. This case can arise in teleoperation and in many manufacturing tasks (i.e. spray-painting, spot-welding, etc.) Second is the case where the targets to be viewed are moving. This can also arise in teleoperation and in other manufacturing tasks (i.e. pick-and-place, part insertion, etc.).

The main difference between these two cases is that in the first case, if a viewpoint is found to be valid at some point during the task, it is guaranteed to be valid with respect to all optical constraints at all times during the task. This is because the functions defining the optical constraints only depend on the target feature locations and the sensor parameters, and not on the positions or orientations of obstacles in the environment. This fairly obvious, but important property allows us to ignore changes in the optical constraints over time and focus only on changes in the geometric parameters, i.e. the visibility constraint.

The second case is more difficult because it requires an examination of how changes in the position and orientation of the target features effect the optical parameters, particularly focus and resolution. However, if the viewpoint is considered in terms of a coordinate frame attached to the feature set, the target can always be considered stationary with the entire environment considered as moving. The only limitation is that the entire feature set must be moving as a single rigid body, i.e. features can not move independently. While extremely important, independently moving features are not yet handled in this work, although it is being examined as part of ongoing research.

To summarize, the exact problem we are dealing with is one in which an accurately movable camera is being used to monitor a task. In this task, the actual target we are monitoring does

point we are actually referring to the *generalized viewpoint* mentioned earlier.

not move, but other objects in the environment, such as a robot arm, or other mechanical parts, move in a way which is known *a priori*. The problem is to find where to place the camera, and when and where to move the camera, so that at all times during the task, we have a good viewpoint for monitoring the task.

4 A Naive Approach

At a first glance, it may seem that the dynamic sensor planning problem can be solved trivially. The naive algorithm for computing a series of viewpoints is as follows:

1. Compute a viewpoint for the initial state of the system, considering all obstacles in the environment as they are before any motion takes place.
2. At every time interval Δt , test the current viewpoint against the model of the changed environment.
3. If, at some instant t_n , the viewpoint is found to be invalid due to the movement of obstacles, compute a new viewpoint based on the current state of the model, and go back to step 2.

There are several problems with this approach. First, it makes no attempt to reduce the number of sensor placements required. Second, a viewpoint is used up until the moment it becomes invalid, or at least up until the point at which the margin for error becomes very small. This defeats the purpose of MVP, which is to find a viewpoint which has as large a margin for error as possible. Worse, by the time a viewpoint is deemed unacceptable, due to errors in sensor placement, etc., the viewpoint may have been invalid for some time.

The basic problem is that this technique does not use knowledge of the motion in computing viewpoints which will be valid for a long period of time. It is conceivable that a new viewpoint will be needed at every Δt , since objects are moving in unaccounted for paths. A better approach, such as the one presented below, uses its knowledge of how objects in the environment move to plan better viewpoints.

5 Overview of Our Approach

The approach being taken is a *Temporal Interval Search* method, which is based on the use of swept volumes. The geometric models of the moving objects are swept through their paths to compute the regions in space which, during some interval, are occupied by some moving object in the environment. The MVP algorithms are then run using the swept volumes for the occluding bodies as opposed to the actual models, thus reducing the dynamic sensor planning problem to a static problem. If no viewpoint is found considering these swept objects over a time interval, a temporal interval search is performed to find the largest time intervals which can be monitored by a single viewpoint. This allows us to plan a series of viewpoints and the times at which they become feasible.

Given that we have an object O whose motion is known over a time interval T , we define $\mathcal{T}(T, O)$ to be the volume swept out by O during T . For example, in 2 dimensions, if O is an axis-aligned unit square moving one unit per second in the positive x direction, and T is 3 seconds, $\mathcal{T}(T, O)$ is a 1×4 square. The key to using swept objects for sensor planning (or, in fact, for any collision avoidance problem) is that in planning around an obstacle given by $\mathcal{T}(T, O)$, you guarantee that you have avoided the actual obstacle O at any instant in interval T . This observation was made by Cameron in [Cameron, 1984] for the "clash detection" (robot collision avoidance) problem.

Let V represent visibility volume for $\mathcal{T}(T, O)$. V is the set of all points (in 3-space) which give views of the target which have no obstructions (due to O) for the entire time interval T . If V is a null volume, there is no single viewpoint which would be valid for all of T . Even if V is not null, there is no guarantee that there are viewpoints within V which satisfy the optical constraints of MVP.

A possible problem when using swept volumes for collision avoidance type problems is that sweeping an object discards all information regarding where the object is at any particular moment. We present a technique for recovering sufficient temporal information to plan sensor locations. If using V as a visibility volume, MVP is unable to find a viewpoint which meets

all constraints, we conclude that T is too large an interval to plan a single viewpoint for, given the motion of O . We have no information concerning when any particular viewpoint becomes invalid; we only know that we can not find a single viewpoint which is valid for the entire interval. Recomputing $T(T, O)$ for a shorter time interval T will yield a smaller obstacle, a larger V , and MVP may now be able to find a viewpoint.

We can now present the algorithm formally. Assume we have a polygonal target τ which we wish to monitor during the time interval $T = [t_0, t_n]$. During T , there is a set of known obstacles O_0 through O_m , which move in known paths. The goal is to plan a single viewpoint valid for the entire interval, if such a point exists, or to determine a sequence of viewpoints which, when executed at the appropriate times, allow the features to be monitored for the entire interval.

Temporal Interval Search

1. Compute $T(T, O_i)$ for each of the m obstacles.
2. Use MVP to compute a viewpoint using $T(T, O_0)$ through $T(T, O_m)$ as well as all stationary objects in the environment as the set of potential occluding bodies.
3. If MVP can successfully find a viewpoint, use this viewpoint for the entire time interval T .
4. If no such viewpoint is obtainable, divide the time interval in half yielding $T_1 = [t_0, t_{n/2}]$. Go back to step 1 using interval T_1 .
5. If the entire time interval T has been planned, we are finished. If not, go to step 1 using the remaining portion of the the original interval T .

Note, this is not strictly a binary search. Step 4 above only looks at the first half of the time interval, i.e. $T_1 = [t_0, t_{n/2}]$. The algorithm searches for the endpoint of the first time interval for which MVP can find one viewpoint. It does this by examining $[t_0, t_n]$, then $[t_0, t_{n/2}]$, $[t_0, t_{n/4}]$, and so on. Once a single viewpoint is found for, say, the interval $[t_0, t_i]$, step 5 sees to it that the interval $[t_i, t_n]$ is examined. If no viewpoint is found for this whole interval, $[t_i, t_{i+(n-i)/2}]$ is examined, and so on, until a single viewpoint is found for, say, the interval $[t_i, t_j]$. This process

continues until a viewpoint has been found which is valid until t_n . The *critical times* are the endpoints of the intervals, i.e. the times at which the sensor must be moved.

The computation of swept volumes is central to this algorithm. Depending upon the format in which the motion is known, the computation of swept volumes may not be expensive. If piecewise linear translational motion is all that is allowed, then the computation of swept volumes is certainly tractable [Weld and Leu, 1990]. However, if more general types of motion are allowed, as in the motions which would be executed by a typical articulated manipulator (rotations in particular), the exact computation of swept volumes is more expensive, but not impossible. Unfortunately, sweeping is not closed over the set of polyhedra when rotational motion is permitted. An articulated robot arm moves strictly in rotations about its joint axes, so the resulting swept volumes are not polyhedral (they would contain circular arcs, spherical patches, and other curved surfaces). These objects would not be useable in MVP. Korein gives an algorithm for computing polyhedral approximations [Korein, 1985] of the swept volumes formed by the motion of articulated robot links. These techniques can be used to simplify the computation of the swept volumes.

Strictly speaking, MVP directly computes volumes of occlusion, not volumes of visibility. In theory, the complement of a volume of occlusion is a volume of visibility. In practice, the complement of a volume of occlusion with respect to the workspace of the manipulator planning the sensor yields the usable visibility volume. In the current dynamic sensor planning implementation, instead of computing a swept volume and then computing its occlusion volume, we compute a set of volumes of occlusion at discrete points along the trajectory. These volumes of occlusion are then unioned to form the volume of occlusion for the entire interval. This is possible because the volume of occlusion generated by the union of a set of obstacles (for viewing a particular target) is equal to the union of the volumes of occlusion generated by each obstacle. One benefit of this approach is that subdivisions of the time interval do not require recomputing new swept volumes. Instead, the appropriate subset of the

instantaneous occlusion volumes can be unioned to approximate the volume of occlusion for any given interval.

6 Realization of the Viewpoints

The result of the temporal interval search will be a set of viewpoints and critical times at which to execute them. However, an explicit representation of time is not required for the temporal interval search, in which case the critical times are not times at all but, rather, critical *events*. If, for example, the motions of a robot have been planned as a series of joint-space moves, the critical events would be joint angle values. If the motion was planned in cartesian space, the critical events would be cartesian positions. Finally, if the robot motion was planned on some global time scale (perhaps avoiding other moving obstacles), the critical events would be actual times on this scale. As long as at task execution time there is a way to determine when the critical events arise, (i.e. by waiting for the robot to be within some distance of the prescribed position), the viewpoints can be realized.

7 Experimental Results

We have modelled our laboratory environment using a CAD system (see figure 1). The model includes two PUMA 560 robots and the object to be monitored during the task. The first robot (I) executes tasks, while the second robot (II) has a camera mounted on it. In the simulated experiment, robot I passes over the object as if it were performing an operation on it, such as spray-painting. During the task, robot II needs to monitor a feature inside the object. A CAD model of the object and the feature is shown in figure 2. The target (i.e. the feature to be viewed) is the top face of the inner cube.

In order to compute viewpoints for monitoring robot I's task, we need to compute the visibility volume for the object as the robot moves in the vicinity of the object, i.e. the volume from which the object is visible during the entire task. In other words, we need to compute the visibility volume for $T(\text{TaskIntervalRobotI}, .)$ The visibility volume is computed by first computing the volume of occlusion, and subtracting it from the reachable work-space of robot II, in order to pre-

vent the computation of a viewpoint which is either unreachable or has an occluded view. The volume of occlusion is approximated using the discrete union algorithm described earlier.

In the experiment, the robot model is stepped through a series of positions along its planned trajectory. At each step, the volume of occlusion is computed as in the static sensor planning problem. The individual volumes of occlusion are unioned together to form the volume of occlusion for the entire trajectory. In this way, we approximate the volume of occlusion for $T(\text{TaskInterval}, \text{RobotI})$ without explicitly computing $T(\text{TaskInterval}, \text{RobotI})$. In figure 4 we show a discrete approximation to the volume swept out by Robot I during its task (i.e. $T(\text{TaskInterval}, \text{RobotI})$). The volume of occlusion resulting from this motion is shown in figure 5. The volume of occlusion resulting from the walls of the part (i.e. due to self-occlusions) is shown in figure 3. These two volumes were unioned to form the total volume of occlusion.

An approximation to the workspace of *Robot II*, the camera-carrying robot, (called the robot's *reachability* volume) was generated. The total occlusion volume was subtracted from this reachability volume giving the reachable/visible volume. This volume, which contains all points in space where the robot can position the camera such that the target can be seen without occlusion, was used in the optimization stage of MVP in order to compute a viewpoint.

Since MVP was unable to find a valid viewpoint for the entire task, the temporal interval search was used to find subintervals for which we can find valid viewpoints. Instead of recomputing the swept volumes for each subinterval examined, the discrete approximation allows us to union the appropriate subset of volumes of occlusion. The subintervals found for this task are shown in figures 6 and 7. The generated volumes of occlusion due to the robot's motion during each sub-interval are shown in figures 8 and 9. These volumes were again unioned with the self-occlusion volume and subtracted from the reachability volume forming the volumes of reachability/visibility shown in figure 10 and 11. These volumes were used in the optimization, and MVP was able to compute a viewpoint for

each interval. Simulated views from these viewpoints are shown in figures 12 and 13.

8 Motion Planning and Moving Sensors

In this section we describe some alternate ways of examining the both the static and dynamic sensor planning problems. The observations and discussions of this section are the motivation for additional research which is currently being carried out.

One can view the static sensor planning problem as a configuration space problem. Using this view, the sensor's possible configurations are described by the generalized viewpoint. The valid configurations are bounded by the constraining hypersurfaces in the 8-dimensional parameter space of the generalized viewpoint. However, the combination of the highly nonlinear fashion of the sensor constraining equations, plus the high dimensionality of the generalized viewpoint, standard techniques for searching configuration spaces appear to be unpractical. This is one of the reasons why MVP takes a numerical optimization approach to searching the sensor's parameter space. However, the configuration-space analogy will be useful in motivating other ideas below.

Dynamic sensor planning is to static sensor planning what path-planning with stationary obstacles is to path-planning with moving obstacles. Erdmann and Lozano-Perez [Erdmann and Lozano-Perez, 1987] proposed a configuration space-time for solving such problems in two dimensions. They presented two approaches, one for translating polygons and one for two-link articulated planar arms. Their approaches focused on the efficient construction of slices of configuration space-time. The slices were chosen so as to include easily computable time-varying constraints, simplifying the search from the start configuration to the goal configuration.

In dynamic sensor planning with stationary targets, the only constraints in configuration space which move are the boundaries of the visibility volume. Even if the obstacles are only allowed restricted classes of motion, their volumes of occlusion not only move but warp, due to the

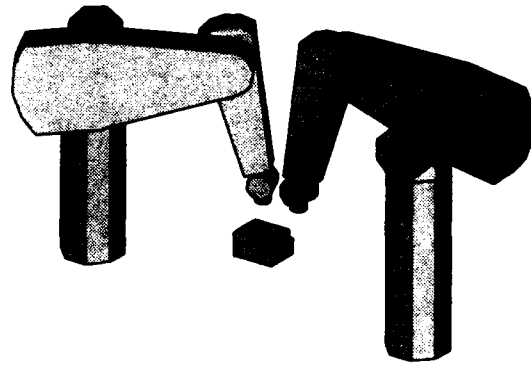


Figure 1: CAD Model of the environment.

fact that the volume of occlusion between an object and a target depends on the relative orientation of the two. Thus, the constraints which are moving in configuration space-time are non-rigid. This makes it very difficult to determine a convenient way of slicing a configuration space-time.

Another way of viewing the dynamic sensor planning problem is to segregate the positioning of the sensor from the orienting and adjusting of the sensor. This allows the computation of a 3-dimensional region from which all constraints can be met (i.e. the projection into 3-space of the set of valid 8-dimensional sensor configurations). The moving polyhedral volumes of occlusion generated by the moving obstacles in the environment can be considered as obstacles which the sensor must avoid while moving in the free-space. This reduces the sensor planning problem to that of keeping a single point away from the boundaries of a set of moving polyhedra. Then, after the sensor path through 3-space has been planned, the other 5 (optical) parameters can be planned accordingly.

This suffers from the same problem as the previous approach, namely that the set of moving polyhedra (the volumes of occlusion), are non-rigid bodies. Although moving polyhedra have been modelled and examined (i.e. [Canny, 1986, Cameron, 1984]), non-rigidly moving bodies have not been examined in detail. It appears that an examination of how the volumes of occlusion change with respect to movements of the obstacles will allow these approaches to be more useful and appears very promising for future research.

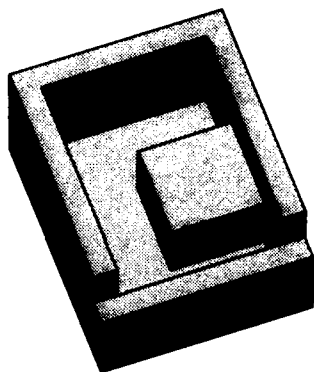


Figure 2: CAD Model of the part to be viewed. The target itself is the top face of the inner cube.

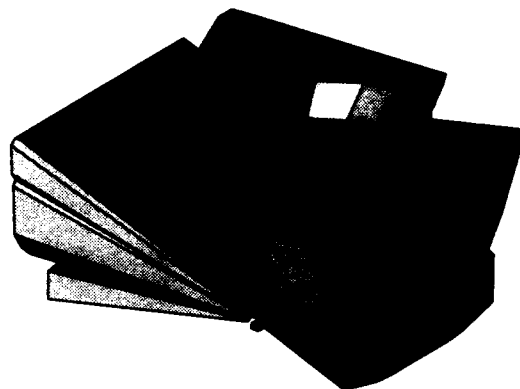


Figure 5: Volume of occlusion caused by the robot's motion during the entire task.

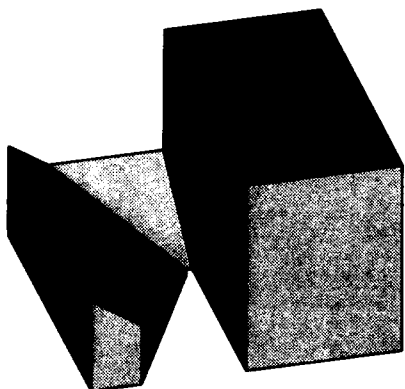


Figure 3: Volume of occlusion caused by other features on the object itself (i.e. self-occlusions).

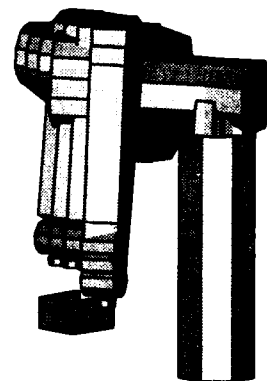


Figure 6: First task interval.

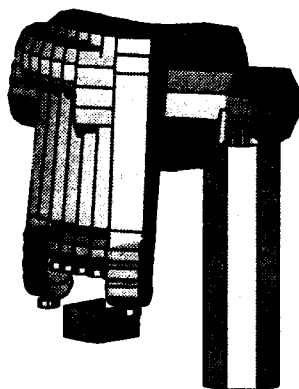


Figure 4: Swept Volume showing the robot's motion over the entire task.

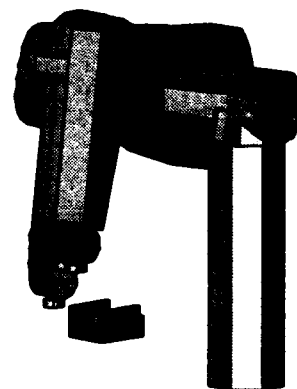


Figure 7: Second task interval.

9 Conclusion

In conclusion, we have successfully extended our MVP system to plan sensor locations in a time-varying environment. This is notable in that to the best of our knowledge, motion has not been widely addressed in the sensor planning literature. The use of swept volumes which provides

a useful way to extend static planning problems to dynamic domains. We have presented a convenient way to recover enough temporal information from swept volumes to use them in planning tasks. Our immediate research plans are to bring the results of this paper into our laboratory and execute the task with the planned viewpoints. Also, we will be examining the alternative sweeping techniques presented to see if

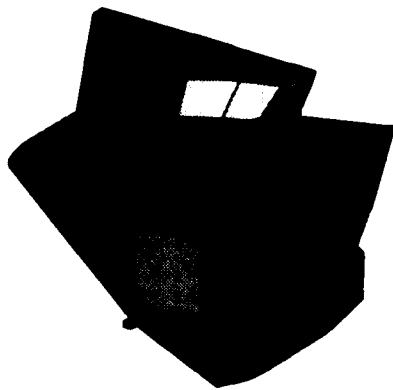


Figure 8: Occlusion due to the robot's motion during the first task interval, shown with object.

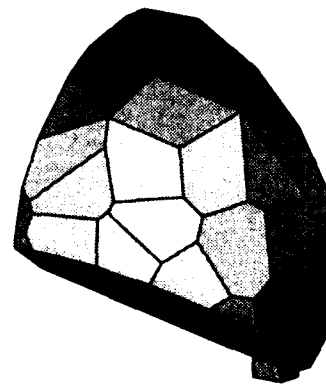


Figure 11: Intersection of reachable and visible volumes for second task interval

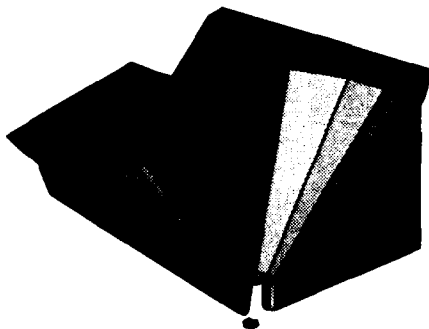


Figure 9: Occlusion due to the robot's motion during the second task interval, with object.

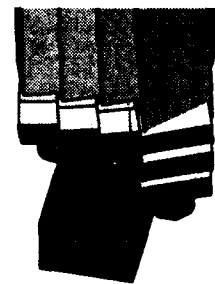


Figure 12: Simulated view from first computed viewpoint.

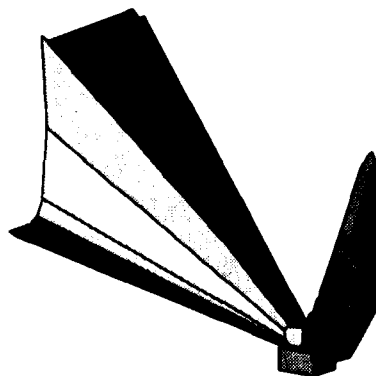


Figure 10: Intersection of reachable and visible volumes for first task interval

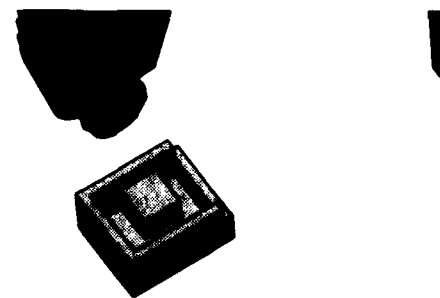


Figure 13: Simulated view from second computed viewpoint.

they offer any performance improvements.

There are several open issues in dynamic sensor planning. There is work to be done in computational geometry to characterize the changes in a volume of occlusion as the target and occluding bodies move with respect to each other. A

similar characterization of how the optical constraints vary with the target's motion is also important. Finally, it is hoped that these various characterizations can be combined to plan a continuous path through the sensor's parameter-space, rather than computing a series of viewpoints and critical times. This would complete

the analogy between sensor planning and configuration space-time based motion planning, and allow more useful solutions to be found to dynamic sensor planning problems.

Acknowledgements

This work was supported in part by DARPA contract DACA-76-92-C-007, NSF grants IRI-86-57151 and CDA-90-24735, North American Philips Laboratories, Siemens Corporation and Rockwell International.

References

- [Abrams and Allen, 1991] Steven Abrams and Peter K. Allen. Sensor planning in an active robotic work cell. In *DARPA 1992 Image Understanding Workshop. Also in Proceedings SPIE Intelligent Robotic Systems Conference on Sensor Fusion IV: Control Paradigms and Data Structures*, Boston, MA, November 1991.
- [Cameron, 1984] Stephen Alan Cameron. *Modelling Solids in Motion*. PhD thesis, Department of Computer Science, University of Edinburgh, 1984.
- [Canny, 1986] John Canny. Collision detection for moving polyhedra. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(2):200-209, March 1986.
- [Cowan and Bergman, 1989] C. K. Cowan and A. Bergman. Model based synthesis of sensor locations. In *Proceedings 1989 IEEE International Conference on Robotics and Automation*, December 1989.
- [Erdmann and Lozano-Perez, 1987] M. Erdmann and T. Lozano-Perez. On multiple moving objects. *Algorithmica*, 2(4):477-521, 1987.
- [Hutchinson and Kak, 1989] S. A. Hutchinson and A. C. Kak. Planning sensing strategies in robot work cell with multi-sensor capabilities. In *Proceedings 1989 IEEE International Conference on Robotics and Automation*, December 1989.
- [Ikeuchi and Kanade, 1989] K. Ikeuchi and T. Kanade. Modeling sensors: Towards automatic generation of object recognition programs. *Computer Vision, Graphics, and Image Processing*, 48:50-79, 1989.
- [Korein, 1985] James Korein. *A Geometric Investigation of Reach*. MIT Press, Cambridge, MA, 1985.
- [Tarabanis et al., 1991a] Konstantinos Tarabanis, Roger Y. Tsai, and Steven Abrams. Planning viewpoints that simultaneously satisfy several feature detectability constraints for robotic vision. In *Proceedings Fifth International Conference of Advanced Robotics*, 1991.
- [Tarabanis et al., 1991b] Konstantinos Tarabanis, Roger Y. Tsai, and Peter K. Allen. Automated sensor planning and modeling for robotic vision tasks. In *Proceedings 1991 IEEE International Conference on Robotics and Automation*, April 1991.
- [Tarabanis, 1991] Konstantinos Tarabanis. *Sensor Planning and Modeling for machine vision tasks*. PhD thesis, Department of Computer Science, Columbia University, October 1991.
- [Weld and Leu, 1990] John D. Weld and Ming C. Leu. Geometric representation of swept volumes with application to polyhedral objects. *International Journal of Robotics Research*, 9(5):105-117, October 1990.

Section XII

Motion and Image Sequences

A feature-based monocular motion analysis system guided by feedback information *

Yong Cheol Kim and Keith Price
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273

Abstract

In the feature-based motion analysis of an image sequence, consistent feature extraction and reliable matching are crucial factors for the motion estimation. Inconsistent feature extraction and erroneous matching are closely related and hard to detect without additional information. In this paper, we address the issues of using errorful data in motion estimation and of using feedback to improve feature extraction and matching in incremental analysis of an image sequence. Thus we use 3-D motion estimation as an aid in generating the data necessary for the motion estimation system itself. Initial noisy correspondence data are continuously refined by removing those parts that do not fit the estimated 3-D motion parameters. The feature extraction of a tracked region is guided by its expected properties which are obtained from the corresponding object in the previous frames. The motion parameters and the environmental depth map are continuously updated with each additional frame. Finally, the surface of environment is reconstructed from a sparse depth map at corners, utilizing the relations among the regions which underlie the corners. Test results for standard real image sequences are presented.

1 Introduction

Recovery of relative motion between the camera and the environment as well as recovery of the environmental structure is an active research area in computer vision. Much of the work has viewed this task purely in mathematical terms - given perfect data, how can we extract 3-D information? In this research, we concentrate on using 3-D motion estimation as an aid in generating the data necessary for the motion estimation system itself.

*This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-90-C-0078. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

Thus we deal with the issues of using errorful data in motion estimation and of using feedback from 3-D estimation to feature extraction and feature matching.

Conventional feature based motion analysis techniques use a sequential framework - feature extraction, establishment of correspondence, estimation of motion parameters and recovery of 3-D structure. This sequential processing of data forces the overall analysis to depend on the integrity of the earlier stages. The reliability of the estimated motion parameters depends on the quality of correspondences, which are affected by the consistency of feature extraction. Feature extraction, feature matching and motion estimation have been studied extensively by numerous researchers, each as a separate research topic.

Establishment of correspondence has been a challenging problem in motion analysis of real image sequences. Sethi [Sethi and Jain, 1987] suggests a tracking method based on the smoothness of motion in image plane. Their work assumes that the number of extracted points remains constant, except for one frame where some points may disappear due to occlusion. Cheng [Cheng and Aggarwal, 1990] uses a 2-stage tracking algorithm for point correspondence in multiple frames. The rule-based second stage inspects the last four frames and updates previous matches by maximizing the smoothness of the 2-D motion.

In recent work on the integration of subsystems into a working motion analysis system, attempts have been made to use feedback from motion. Chandrashekhhar [Chandrashekhhar and Chellappa, 1991] uses predicted 3-D motion for feature correspondence with a partially known structure. Sawhney [Sawhney and Hanson, 1992] uses a predicted mask in the tracking of structure with hypotheses. The main use of motion in these works is to reduce the search space in matching of interest points [Chandrashekhhar and Chellappa, 1991], or in grouping of linear segments [Sawhney and Hanson, 1992].

Little work has been done using feedback of 3-D motion estimation to feature extraction and matching while, in the analysis of an image sequence, the consistency of features extracted over the frames is a crucial factor for a reliable correspondence. In this paper, we present a feedback approach, where feature extraction, matching and motion analysis are performed in cooperative man-

THIS
PAGE
IS
MISSING
IN
ORIGINAL
DOCUMENT

gions and corners speed up computation and increase stability of the matching. Each image in the sequence is segmented into regions (global segmentation), that are matched between adjacent frames. Then, corners computed based on the linear segment approximation of the contours are matched. For convenience, we use RMS (region matching sequence) and CMS (corner matching sequence) to refer to a sequence of matched regions and corners over multiple frames.

A recursive splitting technique [Ohlander *et al.*, 1978] is used for the segmentation of an image which uses the statistics of image attributes (*intensity* for black and white image). The segmentation procedure locates well-separated peaks in the histogram of the image value over a masked area. The image is segmented into regions with a certain range of values of the attribute. Segmented regions are recursively segmented into smaller regions until the size of the region is too small or the attribute is inseparable. After the initial frames in the sequence are segmented this way, the segmentation of regions in the following frames is guided by the expected properties induced from previous matching results.

Matching is performed both in *forward* and in *backward* direction to ensure one-to-one match of features. Relaxation-based symbolic matching [Faugeras and Price, 1981] is used for both region and corner matching. The matching system uses a feature-based symbolic description for its input. For region matching, the properties include average values of the image intensity, size, location and simple shape measures. Relations include adjacency, relative position and near-by. For corner matching, the properties used include position and angular data of the line segments.

The 3-D motion parameters and structure of the matched features are estimated using Chronogeneous analysis technique developed by Franzen [Franzen, 1992], which handles uniform acceleration with constant translation and rotation. Each point need not be visible in all frames, but should be visible in at least 3 frames. The accuracy of the solution depends on the number of frames with steady improvement as the number of frames increases. However, most of the improvement occurs within the first 7 frames with a slight improvement after frame 11. Thus, 7 to 11 frames provide a good compromise between computation time and accuracy.

3 Guidance of motion in feature extraction

To accomplish consistent feature extraction and reliable correspondence, the processing of features is guided by feedback of information in various ways as follows, where motion provides the major guidance information.

1. **Guide in global segmentation** The global segmentation of the current frame is guided by the intensity distribution of the regions extracted from the previous frame.
2. **Guide in feature matching** Matching becomes more stable and faster by limiting the search space of matching along the predicted trajectory.

3. **Refinement of noisy correspondences** Initial noisy correspondence data are gradually refined and linked by 3-D motion. Details are found in [Kim and Price, 1992a].
4. **Guide in local segmentation** In incremental mode, the extraction of a tracked feature is guided by its expected properties (*size, intensity*) induced from the matched regions in previous frames.

3.1 Reference regions

Regions can be *related* in two ways, region matching and corner linking. Matching of segmented regions between adjacent frames becomes disconnected at those frames where similar regions are not extracted or properly matched. The regions in each disconnected RMS are related. Regions in non-adjacent frames become related by linking of the corners generated by the regions.

Figure 2 illustrates all the cases of related regions. An object is segmented into regions 1, 2, 3, 5, 6 in frames 1, 2, 3, 5, 6. In frame 4, only part of the object is segmented into region 4-a and another similar object is segmented into region 4-b. Since region matching fails at frame 4, regions (1, 2, 3) and regions (5, 6) are two sets of regions related by region matching. CMS-a and CMS-b in frames (1, 2, 3) are linked with corners in region 4-a and region 4-b, respectively. CMS-c in frames (5, 6) is linked with a corner in region 4-a. From these relations, regions (1, 2, 3, 4-a, 4-b, 5, 6) are all related. Since only local properties of the contour around the corner are used in the linking process, there can be non-negligible variations in the properties of the related regions. For frame 4, each of the two regions is considerably different from the rest in the other frames.

We need a set of regions that is the most representative of these related regions to be used as the reference in the local feature extraction. First, the set of regions with corners that passed the refinement process are selected. Then they are clustered into sets of consistent regions. The regions in the dominant set are selected as the reference set. The criteria used in the clustering are:

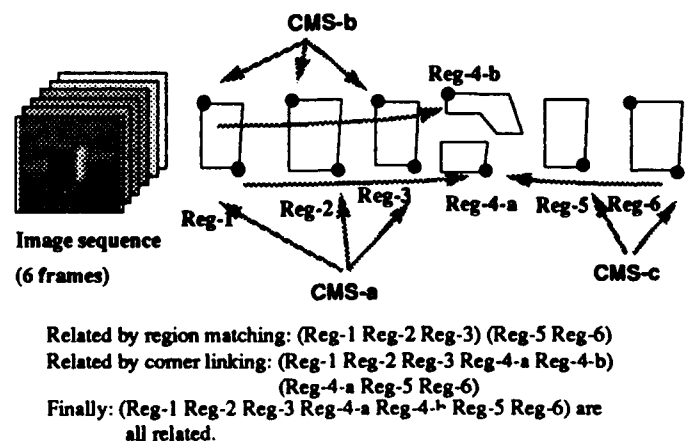


Figure 2: Related regions by corner linking

Criterion 1 Consistent intensity histogram: *The overlap range between the intensity peaks of two regions is larger than α % of the minimum range of the two peaks, where α is usually 40.*

Criterion 2 Consistent contour shape: *The size of the difference of the masks of the two regions after proper scaling and translation to compensate for the motion is less than β % of the average area of the regions, where β is usually 20.*

3.2 Intensive local segmentation

The reference regions focus the segmentation on both the position and shape of the corresponding object. The mask that covers the object of interest is predicted from the shapes of the reference regions. The peak selection process in the intensity histogram of the masked area of the image is guided by the histograms of the reference regions.

The mask for the local segmentation is obtained by scaling and translation of reference regions to compensate for the motion. The motion of a region is represented by that of its corners. When there are several refined CMSs, each CMS produces a predicted mask for the region in the next frame. Consequently, the mask is a union of all the predicted masks from the refined corner sequences of the reference regions.

As the mask fits the desired region more closely, the probability of the region being represented by the dominant peak gets larger. Since the predicted mask is the union of those from each refined corner sequence, usually it covers much larger area than the desired region and thus the histogram of intensity in the mask area usually consists of several peaks, where the desired region is not necessarily represented by the dominant peak. In local segmentation, the peak is selected which overlaps with the intensity peak in the union of the histograms of the reference regions. Thus, the role of the guidance is to pick the correct peak which, otherwise, may be hidden by a larger peak in the mask area. Since the intensity histograms of the reference regions consist of similar peaks, the union of their histograms usually forms a smooth peak. When the region size is small, the union of the histograms with equal weighting may be dominated by an irregular intensity distribution of a large region. To reduce such effects, all the reference regions are given equal weighting in the union of the intensity histograms.

3.3 Merging of global segmentation and local segmentation

We applied guided local segmentation to improve the quality of the globally segmented regions. Comparison of the results from global segmentation and guided local segmentation leads to the following conclusions:

- When the globally segmented regions in an RMS are good in most frames, then guided local segmentation provides some improvement. If, in some of the frames, global segmentation fails to extract a consistent region or fails to generate the desired region at all, then guided local segmentation generates regions with more consistent contour shape, or regions which have not been extracted in the global

segmentation. The role of guided segmentation, in this case, is to fill in the gap of the original RMS.

- When the size of an object is small, the corresponding regions are missing in some of the frames in global segmentation. The resulting RMS includes noisy mismatched regions. When a fine set of reference regions is provided, guided local segmentation is successful in extracting the missing regions. An adaptive minimum size of regions is used in local segmentation, which is a function of the sizes of the reference regions.
- When the RMS from global segmentation is highly noisy, stable reference regions are hard to obtain, and thus the improvement from guided local segmentation is weak.

Thus, the gaps can be filled in by extracting missing regions and a noisy region can be replaced by a locally segmented region. Global segmentation and guided local segmentation are complementary. Guided local segmentation is not used alone because it is focused on an object which has been tracked and is not appropriate in extracting a new region. In incremental analysis, each additional frame is globally segmented and matched. Motion parameters and structure are computed from the matching data and then local segmentation is performed for each RMS that underlies those CMSs that pass the refinement process.

Since the performance of guided local segmentation is affected by the quality of the reference regions, we are conservative in using the results from guided local segmentation. A region is replaced when it is far more consistent with the reference regions. The same criteria 1 and 2 in subsection 3.1 with more strict condition (α , β is 50, 10, respectively) is used for consistency measure. A region in a CMS from global segmentation is replaced by a new region from local segmentation only when the new one meets the criteria and the old one does not.

4 Selection from multiple interpretations of motion

From a set of correspondence data, multiple solutions are generated each associated with a different 3-D MI (motion interpretation) of the features in the scene. The motion analysis algorithm used in our work [Franzen, 1992] is based on a search technique starting from multiple initial guesses. The number of solutions is affected by the quality of the correspondence data with 3-4 solutions generated from reasonably noisy data.

In incremental analysis, several solutions are generated with each new frame. Selection of the correct solution is important since the 3-D motion associated with it guides the processing of the next frame. The fitting error, which is a sum of the differences between the given 2-D positions and the reconstructed 2-D positions in the image, could be a simple measure of the reliability of the solution. However, it is a good criterion only when the quality of the input correspondence data is very good.

The multiplicity of solutions from incremental analysis provides a means to select a good solution by measuring

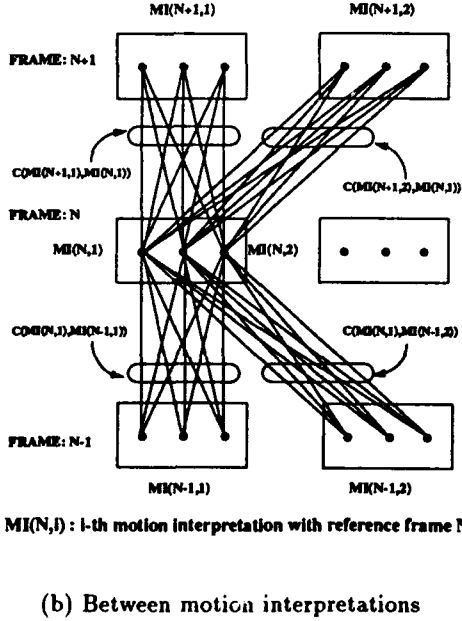
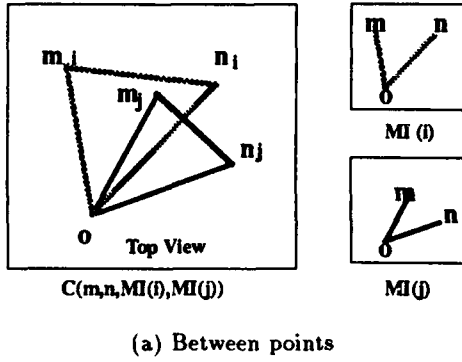


Figure 3: Compatibility measure

the confidence factor of the 3-D MI associated with each solution. We believe that a solution which represents the actual environment is more likely to have coinciding solutions in future frames than a solution which is far from the real situation but happens to fit the given 2-D positions of the correspondence data. The motion parameters from each solution produce a 3-D structure. The confidence factor of a solution is computed from the consistency of the 3-D structure over the sequence of the frames.

Figure 3 illustrates this compatibility measure. First, the confidence factor of a point is computed as the weighted sum of its compatibilities with other points. The compatibility between two points is dependent on the consistency of the relative positions and instantaneous velocity vectors of the two points. Then, the confidence factor of a solution is a weighted sum of the compatibilities of the points in the feature set.

The following definition of compatibility $C_P(m, n, MI(i), MI(j))$ measures the consistency of the

relative positions of point m and n between $MI(i)$ and $MI(j)$ in 3-D space. $C_P(m, n, MI(i), MI(j))$ is scale invariant since it compares the directions of the reconstructed structure.

$$C_P(m, n, MI(i), MI(j)) = \frac{a \cos \theta_m + b \cos \theta_n + c \cos \theta_{mn}}{a + b + c} \quad (1)$$

O : the origin in the object centered coordinate

m_i, n_i, m_j, n_j : the 3-D position of feature m, n in $MI(i)$ and $MI(j)$

θ_m, θ_n : angle of $(m_i, o, m_j), (n_i, o, n_j)$

θ_{mn} : angle between the displacement (m_i, n_i) and the displacement (m_j, n_j)

a, b, c : weighting factor (0.2, 0.2, 0.6 is used)

The confidence factor of an MI is based on the strength of compatibility with other MIs. The compatibility between $MI(i)$ and $MI(j)$ is:

$$C_M(MI(i), MI(j)) = \frac{\sum_{m,n \in F} C_P(m, n, MI(i), MI(j))}{\|F\|^2} \quad (2)$$

where F is the set of common point features and $\|F\|$ is its size.

5 3-D Reconstruction

Motion and structure estimation from corner correspondence data generates a sparse depth map for corners in the image. Since the selected corners are usually from several objects in the image, regular surface interpolation techniques cannot be applied to reconstruct the 3-D surfaces. We build a dense range map using the sparse depth map of corners and the properties (*shape* and *size*) and relations of the underlying region.

Since several corners are generated from a region, the range of a region is represented by the distribution of the depth values of its corners. If the depth estimation is fairly accurate, the local structure of a region could be computed from the depths of several corners along the contour. If the number of corners is large enough, then low-confidence depth values can be eliminated statistically by disregarding those lying at the extremities of the samples as done in [Smith *et al.*, 1992]. In our work, a region usually contains 3 or 4 corners that are matched over the sequences. The number of well-behaved corners that pass through the refinement process is even smaller. The criterion used in our work is the *relative geometric stability* of the estimated 3-D position in eq. 1. We selected the corner that maintains the largest compatibility throughout the frames.

We assume that the objects of interest represented by regions have shallow structure, where the thickness (the difference in depth within the whole structure) is small compared to the depth of the structure [Sawhney and Hanson, 1992]. The depth of the region is represented by that of the corner with the most stable depth value. The 3-D structure for objects of interest are obtained as follows. The shape and size of the region is determined

from its contour. We use 2 models, *cone-type* and *box-type* object for the isolated regions which do not have any descendants in the segmentation. The reference regions from an RMS are labelled at each frame, which depends on the linear approximation of the contour of the region. A region with one corner at the top and two corners in the base is labelled as cone-type. All other isolated regions are labelled as box-type. Consistent labelling is obtained in most cases. When the labelling conflicts among the frames, the model voted by the majority of the reference regions is used as the label.

As stated in section 3, the shape of the contour of a region of an RMS may have large variation from frame to frame. Hence, the region to be used in the surface reconstruction cannot be taken from an arbitrary frame, but should be from the reference regions, which are consistent in contour shape and intensity. The contour of the selected region is scaled and translated to compensate for the motion since the frame number of the selected region varies from RMS to RMS. The height and width of the region is obtained from the bounding rectangle of the contour of the region. The thickness is assumed to be equal to the width of the region, which is not available from the data. A region which has descendants is labelled as a *base* region. A base region is assumed to have flat surface and the orientation of the surface is obtained from an interpolation using the depth values of the well-behaved corners that lie inside the base region.

6 Results

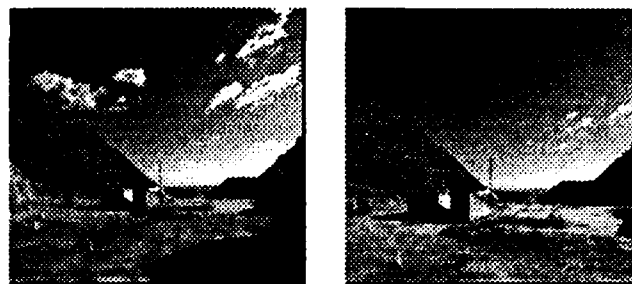
The motion analysis system has been tested for standard sets of real image sequences. We present the results for two image sequences provided by UMASS. The first one, the Rocket field sequence [Dutta *et al.*, 1989], is an outdoor sequence taken by a camera mounted on a vehicle on a terrain, whose motion is dominant translation with some rotational component. Interframe motion is almost constant but has some minor variations.

The second one is the Cone sequence [Sawhney and Hanson, 1992]. The sequence consists of 8 frames and the motion is pure translation along the line of sight. The first and last frames used in the analysis are shown in figure 4.

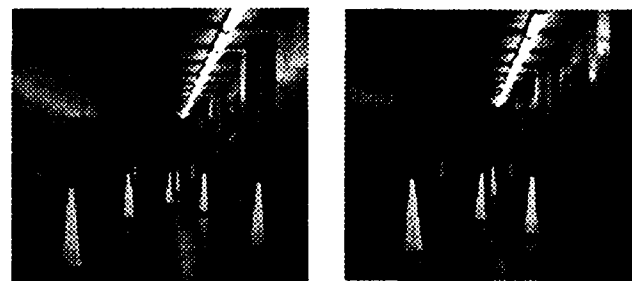
6.1 Guided segmentation

Figure 5 (a) shows a region tracked from frame 1 to frame 6 of the Rocket field sequence. The region represents the front of the building in the image. In global segmentation, the building is extracted into a region from frame 1 to 6 but the corresponding region is not available in frame 7. The building region in frame 7 is extracted by guided local segmentation. In this case, the regions from frame 1 through 6 are so consistent in intensity and shape that all of them become the reference regions. In figure 5 (b), a local mask around the predicted position and the segmented region are shown, which is very consistent to the corresponding regions in frames 1 through 6.

Figure 6 (a) (b) show the image and intensity histogram used in global and local segmentation. Three peaks are found in the intensity histogram in the global



(a) Rocket field sequence



(b) Cone sequence

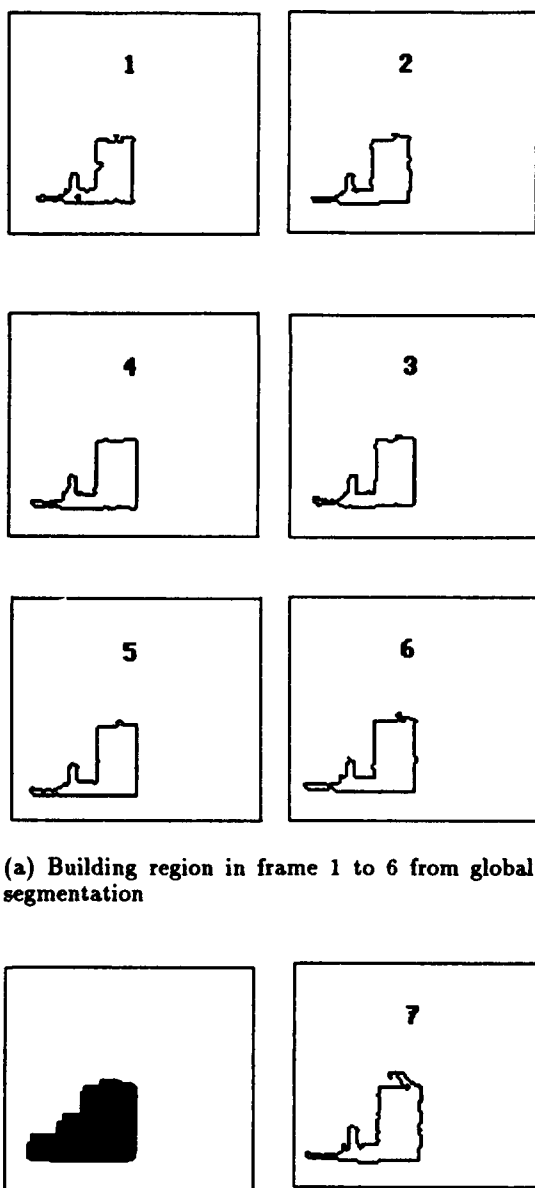
Figure 4: First and last frames of the image sequence

segmentation over the initial mask of the whole image. But none of them represents the intensity values (17 - 41) of the front side of the building. While the area corresponding to each peak continues to be segmented into smaller regions recursively, the area corresponding to the building fails to be extracted into a region. In local segmentation, the mask is represented by the white solid line. The area of the building is represented by the dominant peak in the intensity histogram and is extracted into a region as shown in figure 5 (b). Though the desired region is not necessarily represented by the dominant peak, the guidance from previously matched regions assures the selection of the correct peak if it exists in the histogram.

6.2 Refinement of noisy correspondence

Figure 7 (a) shows the refinement of a CMS from the building region. The extraction of the right upper corner of the building is stable throughout the sequence. The irregular position of the corner at the fourth frame (numbered as 3) is due to the irregular motion of the camera [Dutta *et al.*, 1989]. After the refinement, the corners in frames (0 1 2 4 5 9) are selected and the corner in frame 3 is not used in the motion estimation. In [Kim and Price, 1992b], the gradual improvement of the initial noisy correspondences for the Rocket sequence is described in details.

Figure 7 (b) shows the refinement of a CMS for a



(a) Building region in frame 1 to 6 from global segmentation

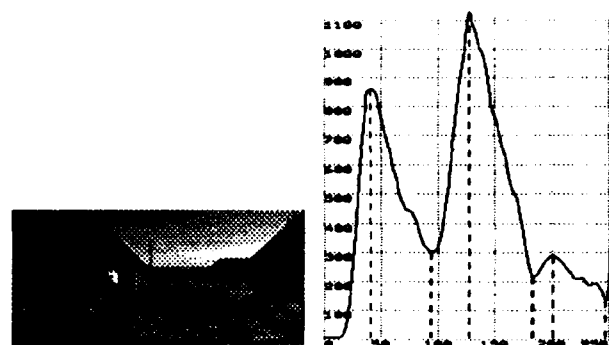
(b) Building region in frame 7 from guided local segmentation

Figure 5: Guided local segmentation

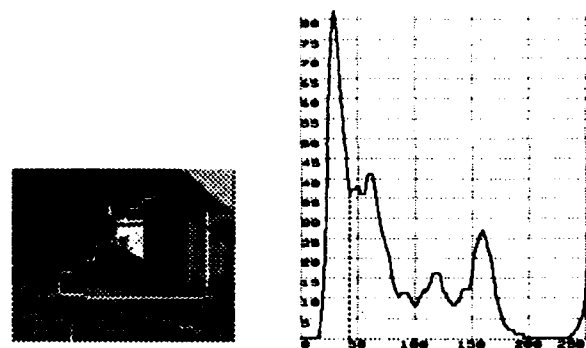
cone in the Cone sequence. For this indoor sequence, the interframe motion of the camera is uniform. Most feature matches are correct. The irregular motion at the third frame (numbered 2) comes from an imperfect segmentation at the frame. The erroneous frame 2 is discarded in the refinement process.

6.3 Incremental analysis

We tested the automated selection of solutions based on eq. 2. The input data are the corner correspondence data from the first 14 frames of the Rocket field Sequence. In each incremental step, the latest 7 frames



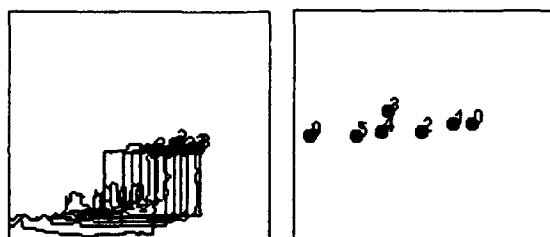
(a) Whole image and histogram in global segmentation



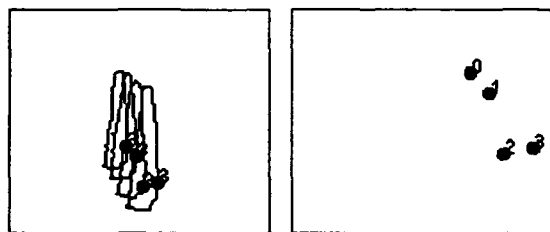
(b) Partial image and histogram in local segmentation

Figure 6: Histograms of intensity

are used with 2 to 4 solutions generated at each step. At each frame, the solution is selected that has the highest compatibility, which is a linear sum of the compatibilities with the selected solutions in the previous frames. In table 1 is shown the compatibility value between the 3-D structure from the selected solution at each frame with the ground truth, which is computed from the motion of the vehicle and the locations of 18 objects in the scene. The number of the tracked CMSs are much larger (393 CMSs) and only those corners are used in the compatibility measure that are common both to the set of the



(a) A building in rocket sequence
Selected corners = (0 1 2 4 5 9)



(b) A cone in cone sequence
Selected corners = (0 1 3)

Figure 7: Refinement of noisy corner trajectories

Frame number	Compatibility	Number of common corners
7	0.795	4
8	0.831	4
9	0.849	6
10	0.883	7
11	0.888	7
12	0.887	7
13	0.889	7
14	0.892	7

Table 1: The compatibility of the best solution from each frame with ground truth values for the Rocket sequence

tracked objects and to the set of objects with ground truth. The compatibility value increases as the frame number increases and the improvement is slow after a value of 0.89 (The maximum possible value is 1.0).

Figures 9 (a), (b) show the reconstructed trajectories and the top view of the objects with ground truth for the Rocket sequence. The top views in the incremental analysis for the eighth, tenth, twelfth and fourteenth frames are shown in figures 9 (c), (d), (e), (f). When compared with the ground truth top view in figure 9 (b), we find that the order of depth is reversed for part of the objects but the estimated motion of the camera is close to the real motion.

6.4 3-D structure

Figure 8 (a) and (b) show the reconstructed trajectories and the top view of the objects for the Cone sequence. The estimated motion of the camera shown as a straight line in the top view is very close to the real motion. The estimated positions of the cones and the trash box agree with the data given in [Sawhney and Hanson, 1992]. Figure 8 (c) shows a rendered view of the reconstructed 3-D structure of the cones and the trash box, seen from a different viewing angle.

7 Conclusion

In this paper, we presented an approach to use feedback in the domain of feature-based motion analysis for a monocular image sequence. We extended the scope of the guidance of the feedback of 3-D motion from refinement of features to feature extraction. The incoming frames are analyzed incrementally with several solutions generated at each frame. We devised a scheme of automatic selection of the correct solution that is based on the relative geometric stability of the estimated 3-D position.

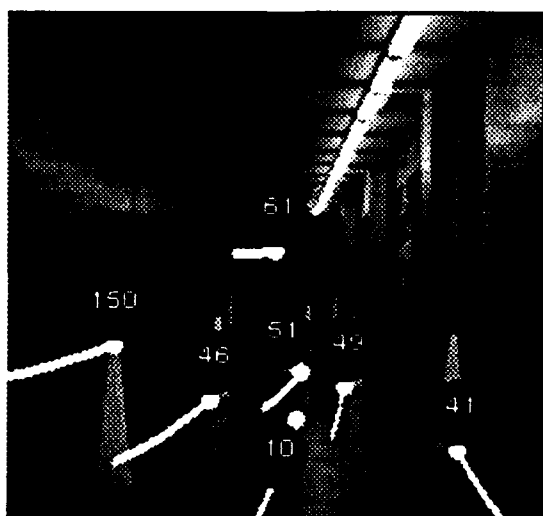
We applied this approach in an automated motion analysis system that is built on hierarchical feature extraction and matching. Standard real image sequences are used as the test set of our system and the results for two image sequences (one outdoor sequence and one indoor sequence) are presented.

We showed that the initial noisy correspondence data are gradually refined and that the extraction of a tracked region is improved when guided by its expected properties obtained from the corresponding objects in the previous frames. 3-D surfaces of the objects are reconstructed using the sparse depth map from the estimates of 3-D point positions and the motion parameters.

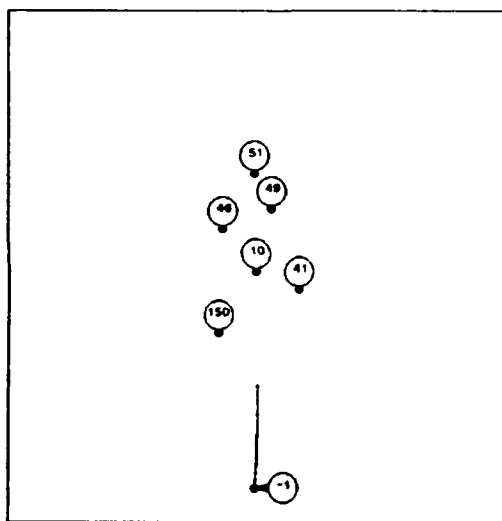
Our motion analysis system is based on very common subsystems. We believe that the idea of using feedback to improve the extraction and matching of features can be applied to other motion analysis system. A drawback of our system is that the refinement of correspondence assumes that most of the objects in the scene belong to one major motion group as is the case with the egomotion of the camera in stationary environments. When there are multiple motion groups none of which are dominant, the result of the refinement process is unpredictable in the present implementation. This problem can be solved using a clustering of the correspondences with hypotheses of multiple motion, which requires a costly computational process.

Acknowledgments

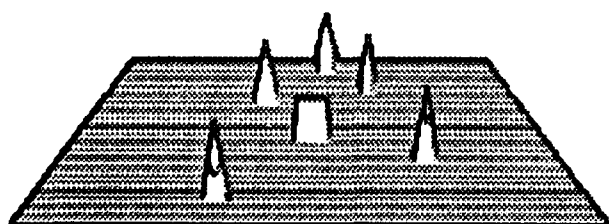
The authors would like to thank Harpreet S. Sawhney at University of Massachusetts, Amherst for providing the Cone image sequence data.



(a) Reconstructed front view



(b) Reconstructed top view where the camera is numbered as -1.



(c) Rendered 3-D view

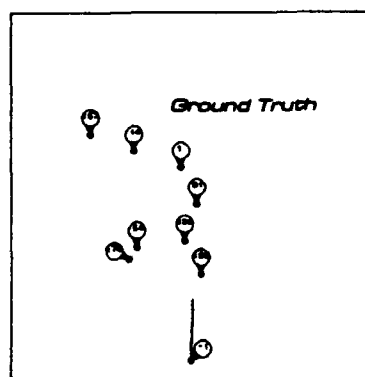
Figure 8: Reconstructed structure for the Cone sequence

References

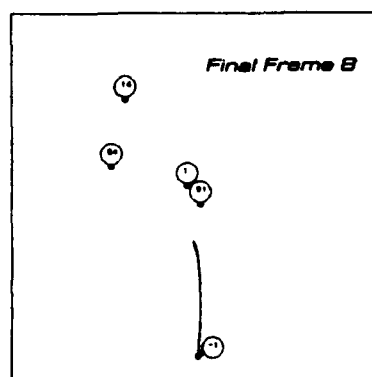
- [Chandrashekhar and Chellappa, 1991] S. Chandrashekhar and R. Chellappa. Passive navigation in a partially known environment. In *Proceedings of the Workshop on Visual Motion*, pages 2-7, 1991.
- [Cheng and Aggarwal, 1990] C. Cheng and J. Aggarwal. A two-stage hybrid approach to the correspondence problem via forward-searching and backward-correcting. In *Proceedings of the International Conference on Pattern Recognition*, pages 173-177, 1990.
- [Dutta et al., 1989] R. Dutta, R. Manmatha, L. Williams, and E. Riseman. A data set for quantitative motion analysis. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 159-164, 1989.
- [Faugeras and Price, 1981] O. D. Faugeras and K. Price. Symbolic description of aerial images using stochastic labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(6):633-642, November 1981.
- [Franzen, 1992] W. O. Franzen. Structure from chronogeneous motion: A summary. In *Proceedings of the DARPA Image Understanding Workshop*, San Diego, CA, January 1992.
- [Kim and Price, 1992a] Y.C. Kim and K. Price. Refinement of noisy correspondence using feedback from 3-d motion. In *Proceedings of the DARPA Image Understanding Workshop*, pages 479-486, 1992.
- [Kim and Price, 1992b] Y.C. Kim and K. Price. Refinement of noisy correspondence using feedback from 3-d motion. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 836-838, 1992.
- [Ohlander et al., 1978] R. Ohlander, K. Price, and R. Reddy. Picture segmentation by a recursive region splitting method. *Computer Graphics and Image Processing*, 8:313-333, 1978.
- [Sawhney and Hanson, 1992] H. Sawhney and A. Hanson. Tracking, detection and 3d representation of potential obstacles using affine constraints. In *Proceedings of the DARPA Image Understanding Workshop*, pages 1009-1017, 1992.
- [Sethi and Jain, 1987] I. K. Sethi and R. Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):56-73, January 1987.
- [Smith et al., 1992] P. Smith, B. Sridhar, and B. Hussien. Vision-based range estimation using helicopter flight data. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 202-208, 1992.



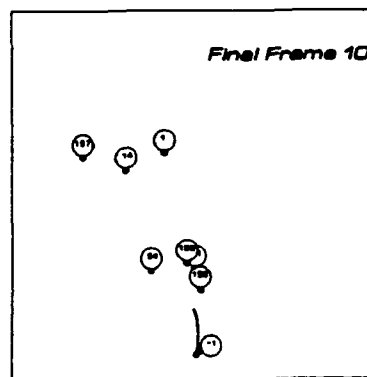
(a) Reconstructed front view



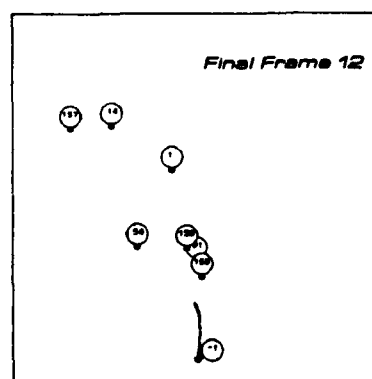
(b) Ground truth view



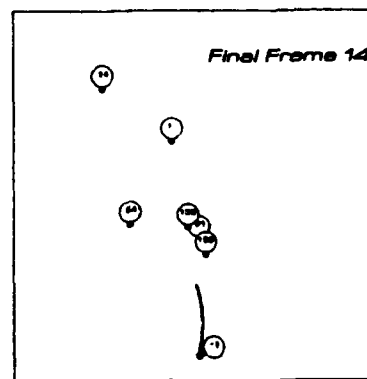
(c)



(d)



(e)



(f)

Figure 9: The top view of estimated structure in incremental analysis for Rocket sequence

Detection and Segmentation of Feature Trajectories in Multiple, Discontinuous Motion Image Sequences*

Srikanth Thirumalai and Narendra Ahuja
Beckman Institute, 405 N. Mathews, Urbana IL 61801.
Email: srik, ahuja@vision.csl.uiuc.edu

Abstract

This paper is concerned with three-dimensional interpretation of image sequences showing multiple objects in motion. Each object exhibits smooth motion except at certain time instants when a motion discontinuity may occur. The objects are assumed to contain point features which are detected as the images are acquired. The problem of estimating initial feature trajectories, in the first two frames, is that of feature matching. As more images are acquired, existing trajectories are extended. Both initial detection and extension of trajectories are done by enforcing pertinent constraints from among the following: similarity of image plane arrangement of neighboring features, smoothness of three dimensional motion and smoothness of image plane motion. As trajectories are estimated, they are segmented into subsets each corresponding to a different object. Both detection and segmentation are formulated as cost minimization problems which enforce appropriate sets of the above constraints. Cost minimization in each case is done using a Hopfield network. Experimental results on several image sequences are shown.

1 Introduction

This paper is concerned with three-dimensional interpretation of image sequences showing multiple objects in motion. Each object exhibits smooth motion except at certain time instants when a motion discontinuity may occur. A common type of motion discontinuity is in motion direction, e.g. when an object undergoes a collision. Between such instants of temporal discontinuity each object exhibits a smooth motion.

The objects are assumed to contain point features which are detected as images are acquired. Trajectory detection begins with the first two frames wherein it amounts to the standard problem of feature matching. As more images are acquired, existing trajectories

are extended. There is little information available for the initial matching of features in the two frames. So any matching constraints are potentially error prone and must be further confirmed against subsequent frames. Such initial matching of features is done based on the similarity of the image plane arrangements of their neighbors. Clearly, this only holds for those neighbors which are detected in both frames, and provided the neighbors do not belong to another neighboring object. Once the first two frames are matched, the initial segment of each feature trajectory is found. There is now more information available for matching of the features in the second frame with those in the third, i.e., for trajectory extension. The extension of trajectories must be consistent with continuation of the three-dimensional motion of the object. However, this is only true when the object is not undergoing a motion discontinuity. Further, this constraint can be applied only if features have been segmented into objects so that three-dimensional motion of an object can be estimated. When the smoothness of the three-dimensional motion cannot be enforced, smoothness of trajectories are constrained to have only two-dimensional smoothness which is possible once the initial trajectories are detected, i.e., beyond the second frame, and which is correct except across temporal discontinuities in motion. Thus, several different constraints are used to detect and extend trajectories, but each of these constraints must be used when it is applicable.

As the images are acquired and trajectories are estimated, they are segmented into subsets each corresponding to a different moving object. This is done by identifying neighboring correspondences and breaking these neighbor relationships if they are found to be very dissimilar.

Both problems, feature matching and trajectory detection, as well as segmentation are formulated as cost minimization problems. The costs are defined in terms of the constraints mentioned above, such that only appropriate constraints are used at any given image location at any given time. Each of the two problems is mapped onto a different Hopfield network which does the cost minimization.

*This work was supported by the Defense Advanced Research Projects Agency and the National Science Foundation under grant IRI-8902728

have been no attempts though, to integrate cost measures from several constraints, as discussed above, to solve more complex scenes with multiple moving objects exhibiting temporal discontinuities in their motion. Also, none of the previous approaches offer any scheme to eliminate wrong matches which result from settling down to a local minimum while performing energy minimization. This makes the approach more robust to variations in scene parameters like inter-frame motion of objects which can vary from 0 to 20 pixels.

Section 2 gives the details of the formulation of feature matching and trajectory detection problems. Sections 3 and 4 describe the mapping of the problems of feature correspondence and trajectory finding onto the Hopfield network. Section 5 describes the algorithm to eliminate wrong correspondences that result from descending to a local energy minimum. Section 6 describes a segmentation scheme to segment the correspondences into groups representing rigid objects. Finally, Section 7 contains results on several image sequences.

2 Feature Matching and Trajectory Detection

2.1 Feature Matching

The problem of feature correspondence deals with finding a match in the current frame, if it exists, for every point in the previous frame. In order to find the correct match, constraints like uniqueness and image plane similarity in the arrangement of neighbors around the points are imposed. The uniqueness constraint implies that a point in the previous frame can match at most one point in the current frame and vice versa. The other constraint implies that the point in the current frame which is the correct match for a certain point in the previous frame should have a similar geometrical arrangement of neighbors around it. These constraints are then incorporated into an energy function in such a way that the energy function attains a minimum when the points are either correctly matched or not matched at all. This energy minimization is done using the gradient descent approach that is implemented using a Hopfield network.

Let the first frame have N_1 points and the second frame N_2 points. We have a matrix of $N_1 \times N_2$ possible match hypotheses where element (i, j) indicates that the i^{th} feature in the first frame matches the j^{th} feature in the second frame. Only a subset of these are correct and these represent the correct matches. A cost is associated with every hypothesis such that a low cost is assigned to a correct hypothesis and a high cost is assigned to a wrong hypothesis. Since the motion of the feature points is bounded by a maximum possible motion, we compute the costs for only those hypotheses where the point in the second frame is within a *circle of interest* of the point in the first frame. All other hypotheses are assigned a fixed high cost. The costs are computed based on the image plane similarity in the arrangement of *neighbors* around the two points consti-

tuting the hypothesis. The neighbors discussed here are the Delaunay neighbors of the points. To determine the cost associated with a match hypothesis, we must try to find a subset of neighbors of the point in the first frame that match a subset of neighbors of the point in the second frame. The similarity in the image plane arrangement of these subsets is used to compute the cost associated with the match. We look for similarity in the subsets rather than the entire set of neighbors because missing feature points and object boundaries cause distortions in the set of neighbors. This cost computation scheme is explained in Section 4.1. The costs associated with all the possible $N_1 \times N_2$ hypotheses are then incorporated into an energy function which when minimized yields N_{12} trajectories of length 2.

2.2 Trajectory Detection

The trajectory detection problem is just an extension of the feature correspondence problem where the trajectories obtained till the previous frame are to be matched to the points in the current frame if such a match does exist. In addition to constraints like uniqueness of a match and image plane similarity in the arrangement of neighbors around points corresponding to a correct match, other constraints like 2-D (image plane) continuity of trajectories and 3-D motion continuity can be imposed.

Consider the problem of extending the N_{12} trajectories obtained thus far to the third frame. The problem can be restated as that of having to match the N_{12} trajectories to the N_3 points in the third frame. In this situation, we have a matrix of $N_{12} \times N_3$ hypotheses. Of these hypotheses, there will be a subset of hypotheses that represent matches between trajectories and points in the third frame that lie within their circles of interest. The costs to be associated with each of these hypotheses have to be determined. All of the other hypotheses (those involving trajectories and points outside their circles of interest) will be assigned very high costs. The costs associated with each hypothesis are representative of the following three constraints.

1. Similarity between the arrangement of neighbors around the last point of the trajectory and the point in the third frame. This is exactly the same as the constraint imposed on point correspondences.
2. Continuity of the 3-D motion computed from the trajectories already known.
3. 2-D continuity of the trajectories.

The extent to which the hypotheses satisfy each of the above constraints is determined separately. This results in three different cost measures that need to be merged before being incorporated into the energy function. The second method is now explained. Having obtained the trajectories till the previous frame, any applicable motion and structure algorithm can be used to estimate the 3-D motion and structure of the objects and these estimates can be used to predict the positions of the

feature points in the current frame. Based on the predicted positions, a cost can be associated with every hypothesis that a trajectory extending upto the previous frame matches a point in the current frame. Before applying any motion model one has to segment the trajectories into groups that correspond to rigid objects. This cost computation scheme is described in detail in Section 4.2.

The third method to compute the cost to be associated with a trajectory and a point in the third frame that lies in its circle of interest is based on the image plane continuity of the trajectory across the frames. This is done as follows. The time axis is collapsed so that the problem of fitting a function to the trajectory becomes a one-dimensional problem. A Lagrange interpolation is done between the points constituting the trajectory and the slope at the last point of the trajectory is computed. Then, the slope of the line segment joining the last point of the trajectory and the point in the current frame that is competing for the match is computed. The cost associated with the hypothesis is then based on the similarity of the two slopes.

We thus have three different cost measures that are associated with each competing hypothesis. A method to merge these three costs and associate a single cost measure with each competing hypothesis is devised.

2.3 Cost Merge Algorithm

Three cost computation schemes have been outlined above. Each method works well in some cases but fails in others. The 2-D geometrical cost computation method might not yield good results at points close to object boundaries because of differences in the motions of the two objects. It could also happen that though a certain point in the current frame is the correct match for a point in the previous frame, there might be no subset of neighbors that match for the two points. The cost computed using the assumption of 3-D motion continuity across frames yields better results than the 2-D method but again fails at object boundaries because the rigidity assumption is violated across boundaries. This method also fails when there is a temporal discontinuity in the motion of objects. Finally, the cost computation scheme based on the 2-D continuity of trajectories gives good results at object boundaries but fails when there is a temporal discontinuity of motion. In a situation in which there is a temporal discontinuity in the motion, only the cost based on the 2-D arrangement of neighbors can be used. Table 1 lists the cases in which the three methods fail or apply.

This suggests that there is a need to merge these costs and determine a single cost measure for a certain match that does not fail at either object boundaries or motion discontinuities. One could suggest many methods to achieve this. In this system we have taken the simple approach of choosing the least of the three costs. In our experiments we have seen that this method works pretty well, and there is no need for a more complex cost merge

scheme. Once the merged cost associated with every hypothesis is computed, they are incorporated into an energy function along with the uniqueness constraints. This energy function is minimized using the Hopfield network.

3 Mapping the feature correspondence and trajectory finding problems onto the Hopfield Network

3.1 Mapping the feature matching problem onto the network

Consider the case in which there are two images of the scene taken at two different time instants, t_1 and t_2 . The feature extractor is run on the two images and two sets of feature points are obtained. Let the first image have N_1 points and the second image have N_2 points. The problem at hand is to find a subset of points, N_{12} in number, in the first frame that has matches in the second frame.

The Hopfield network [6] used for the above problem consists of a two-dimensional array of processing elements (PEs) having N_1 rows corresponding to the N_1 points in the first frame and N_2 columns corresponding to the N_2 points in the second frame. Each PE is essentially a nonlinear amplifier that produces an output v_i which is related to its input u_i by the equation

$$v_i = g(\lambda u_i) = \frac{1}{2}(1 + \tanh(\lambda u_i)) \quad (1)$$

where λ is called the gain parameter. The input u_i to the i^{th} PE is the weighted sum of the outputs of the PEs that are connected to it. The processing element (i, j) represents the hypothesis that the i^{th} point in frame 1 matches the j^{th} point in frame 2. Each processing element has a potential associated with it. This potential corresponds to the quantity v discussed previously and can take on a continuum of values between 0 and 1. The value 1 represents a sure match between the corresponding points in the two frames and the value 0 represents a nonmatch. Any value between 0 and 1 signifies the level of confidence in the match between the corresponding points.

The connections between the processing elements and the weights associated with them depend on the energy function that has to be minimized. The energy function used in this problem has the form

$$\begin{aligned} Energy = & -\frac{1}{2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \sum_{k=1}^{N_1} \sum_{l=1}^{N_2} T_{ij,kl} v_{ij} v_{kl} \\ & - \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} I_{ij} v_{ij} \\ & + \frac{1}{\lambda} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \frac{1}{R_{ij}} \int_0^{v_{ij}} g^{-1}(v) dv \quad (2) \end{aligned}$$

Table 1: Comparative analysis of the effectiveness of the three cost computation schemes

Cost Computation Schemes	Fails	Works
2-D arrangement of neighbors	When no subsets of neighbors match	All other times including when there is a temporal discontinuity in the motion
Continuity of 3-D motion	At object boundaries and temporal motion discontinuities	All other times
Continuity of trajectories	Temporal motion discontinuities	All other times

where, v_{ij} represents the potential of the $(i, j)^{th}$ PE, $T_{ij,kl}$ represents the weight of the link from the $(k, l)^{th}$ PE to the $(i, j)^{th}$ PE, I_{ij} represents the bias input to the $(i, j)^{th}$ element and R_{ij} represents the resistance seen at the input of the $(i, j)^{th}$ PE.

Ignoring the integral term in the previous equation which was present due to the introduction of an intermediate variable u_{ij} , the energy function can also be written, specifically for the problem of feature correspondence, as

$$\begin{aligned}
Energy = & \frac{A}{2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \sum_{k=1, k \neq j}^{N_2} v_{ij} v_{ik} \\
& + \frac{B}{2} \sum_{j=1}^{N_2} \sum_{i=1}^{N_1} \sum_{k=1, k \neq i}^{N_1} v_{ij} v_{kj} + \frac{C}{2} \left(\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} v_{ij} - N_{12} \right)^2 \\
& + \frac{D}{2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \sum_{k=1, k \neq j}^{N_2} (Cost(i, j) - Cost(i, k)) v_{ij} v_{ik} \\
& + \frac{E}{2} \sum_{j=1}^{N_2} \sum_{i=1}^{N_1} \sum_{k=1, k \neq i}^{N_1} (Cost(i, j) - Cost(k, j)) v_{ij} v_{kj} \quad (3)
\end{aligned}$$

Each term in the above equation has a physical explanation that is outlined below. The first term deals with Row Inhibition. This term ensures that when the network stabilizes, there is at most one PE in each row that has a potential of 1 whereas all of the other elements have value 0. The constant A determines the relative importance this term is given w.r.t. the other terms. The second term deals with Column Inhibition. This is the column analog of the first term. When the network stabilizes, at most one in each column has a potential of 1. Again the constant B decides the relative importance this term is given w.r.t. the other terms. The first two terms in the above equation enforce the concept of uniqueness of a match, i.e., a point in the first frame can match at most one point in the second frame and vice versa. Since both of these terms are equivalent, we generally give them equal importance, i.e., we have $A = B$. The third term in the energy equation deals with Global Inhibition. This term is minimum, i.e., 0, only when the total number of 1's in the array

is N_{12} . This term ensures that there are approximately N_{12} matches obtained when the network stabilizes. At the energy minimum, in most cases, we will not have exactly N_{12} matches but some number that is close to it. In this problem we set N_{12} to $\min(N_1, N_2)$. The fourth term deals with Cost Based Row Inhibition. For a certain point in the left frame, all of the points in the second frame compete for a match. If a certain point in the second frame has a lower cost associated with it than another point, then it tries to reduce the potential of the other hypothesis by issuing an inhibitory signal. This reduces the potential of the other PE. In this term again D determines the relative weight that this term has in the final expression. Analogously, the last term deals with Cost Based Column Inhibition. Ideally the last two terms should also be 0 when the network stabilizes on a solution.

Comparing Equations (2) and (3) we obtain

$$\begin{aligned}
T_{ij,kl} = & -A\delta_{ik}(1 - \delta_{jl}) - B\delta_{jl}(1 - \delta_{ik}) - C \\
& -D(Cost(i, j) - Cost(k, l))\delta_{ik}(1 - \delta_{jl}) \\
& -E(Cost(i, j) - Cost(k, l))\delta_{jl}(1 - \delta_{ik}) \quad (4)
\end{aligned}$$

where $\delta_{mn} = 1$ if $m = n$ and $\delta_{mn} = 0$ if $m \neq n$. We also obtain

$$I_{ij} = CN_{12} \quad (5)$$

We use the gradient descent method to approach the energy minima. The equation for gradient descent can be written as

$$\frac{du_{ij}}{dt} = -\frac{\partial(Energy)}{\partial v_{ij}} \quad (6)$$

which yields

$$\frac{du_{ij}}{dt} = \sum_{k=1}^{N_1} \sum_{l=1}^{N_2} T_{ij,kl} v_{kl} - \frac{u_{ij}}{\tau_{ij}} + I_{ij} \quad (7)$$

where $\tau_{ij} = R_{ij}C$, the time constant. In our formulation of the problem, we have assumed that the time constant is the same for all processors. This does not affect the solution but only decides the rate of convergence.

A digital simulation of this system requires that we integrate these equations numerically. For a sufficiently

small value of Δt , we can write

$$\Delta u_{ij} = \left(\sum_{k=1}^{N_1} \sum_{l=1}^{N_2} T_{ij,kl} v_{kl} - \frac{u_{ij}}{\tau} + I_{ij} \right) \Delta t \quad (8)$$

The values of u_{ij} can be iteratively updated according to the following rule.

$$u_{ij}(t+1) = u_{ij}(t) + \Delta u_{ij} \quad (9)$$

The final output potential of the PE is given by

$$v_{ij} = g(u_{ij}) = \frac{1}{2}(1 + \tanh \lambda u_{ij}) \quad (10)$$

If we substitute Equations (4) and (5) into 8 we obtain the following result:

$$\begin{aligned} \Delta u_{ij} = & \left[-\frac{u_{ij}}{\tau} - A \sum_{k=1, k \neq j}^{N_2} v_{ik} - B \sum_{k=1, k \neq i}^{N_1} v_{kj} \right. \\ & - C \left(\sum_{k=1}^{N_1} \sum_{l=1}^{N_2} v_{kl} - N_{12} \right) \\ & - D \sum_{k=1, k \neq j}^{N_2} (Cost(i, j) - Cost(i, k)) v_{ik} \\ & \left. - E \sum_{k=1, k \neq i}^{N_1} (Cost(i, j) - Cost(k, j)) v_{kj} \right] \Delta t \end{aligned} \quad (11)$$

Equations (9) and (11) describe the dynamics of the network. Now only the initial values of the potentials of the PEs (v_{ij} for the $(i, j)^{th}$ PE) have to be specified. Having done this the network could be allowed to evolve in time until it attains a steady state, i.e., a stage where the outputs of the PEs do not change. The output of each PE is initialized to $v_{ij} = 1.0 - Cost(i, j)$. The initial output could then be viewed as the probability that the corresponding hypothesis is true. The network can then be allowed to evolve in time until it stabilizes. The process of evolution is auto-association. The advantages of this initialization scheme are that a fewer number of iteration steps are needed and the solution obtained is better than that obtained using random initialization. After the network stabilizes, all of the PEs have outputs equal to either 0 or 1. Those that have outputs 1 have been identified as the correct hypotheses while those that have outputs 0 have been identified as the wrong hypotheses.

3.2 Mapping the trajectory detection problem onto the network

The problem of establishing the correspondence between the trajectories computed until the previous frame and the points in the current frame is similar, in formulation, to the problem of feature correspondence between two frames. The only difference is in

the method used to calculate the costs associated with every competing hypothesis. In the case of feature correspondence between two frames, we rely only on the cost based on the image plane similarity in the arrangement of neighbors around the point pairs constituting the hypotheses. In the case of correspondence between the trajectories and points, there are three cost measures. These are based on the image plane similarity in the arrangement of neighbors around the last point of each trajectory and the point in the current frame, continuity of 3-D motion, and continuity of trajectories. The details of the implementation of these three cost computation schemes are given in the next section.

4 Cost Computation

4.1 Two-Dimensional Geometrical Cost

This cost is computed to reflect the similarity in the image plane arrangement of the Voronoi neighbors of the two points constituting a candidate match. Let the Voronoi neighbors of point i in the previous frame be (a, b, c, d, e) and that of j in the current frame be (a', b', c', d') . We have to determine which subset of the lines (ia, ib, ic, id, ie) matches a subset of lines (ja', jb', jc', jd') . This is done using a two dimensional Hopfield network similar to the one described in Section 3. When this match is determined, the similarity in the image plane arrangement of the neighbors can be estimated. The cost function needed to initialize this network is based on the similarity in the lengths and orientations of the lines competing for a match. This function is defined as

$$\begin{aligned} \text{simil} = & F(C_1 - C_2(\Delta \text{length}_{\text{actual}} - \Delta \text{LENGTH}) \\ & - C_3(\Delta \text{orientation}_{\text{actual}} - \Delta \text{ORIENTATION})) \end{aligned} \quad (12)$$

where F is a non-linear function defined by $F(x) = x, x \geq C_4$ and $F(x) = C_4, x < C_4$, and $C_4 = -(C_1 + C_2 \Delta \text{LENGTH} + C_3 \Delta \text{ORIENTATION})$. Here $\Delta \text{length}_{\text{actual}}$ and $\Delta \text{orientation}_{\text{actual}}$ are the actual absolute differences in the length and orientation between the lines constituting a match hypothesis. ΔLENGTH and $\Delta \text{ORIENTATION}$ are constants to be provided a priori. The similarity measures are normalized and the cost for each match is computed as $\text{cost} = 1.0 - \text{similarity}$.

After the network computes the right subset of matched lines, then for every matched line pair ia and ja' , the quantity $1.0 - \text{dotproduct of } \vec{ij} \text{ and } \vec{aa'}$ is computed. The minimum of the above quantity over all the matched pairs is the final two-dimensional geometric cost.

4.2 Motion model cost

The correspondences obtained between the first two frames can be segmented into groups belonging to different rigid objects as described in Section 6. A motion

model can be used to predict the positions of the feature points in the next frame. Depending on the number of correspondences available, suitable motion models could be used. We have used the local translation model where we fit the model to a correspondence and its Voronoi neighbors. Based on the 3D motion computed, the positions of the feature points in the current frame are predicted such that each trajectory till the previous frame has one predicted feature point in the current frame associated with it. The cost of matching a trajectory till the previous frame to a point in the current frame is done by determining the similarity in the image plane arrangement in the neighbors of the predicted points and the actual points in the current frame. This is done in a manner similar to Section 4.1. As the length of the trajectories increases, more complex motion models could be used.

4.3 Cost based on the continuity of trajectories

Consider a trajectory of length $(n + 1)$ extending from the first frame to the $(n + 1)^{th}$ frame. Let the image plane co-ordinates of the points constituting the trajectory be $(X_0, Y_0), (X_1, Y_1), \dots, (X_n, Y_n)$. We fit a function to this set of points using Lagrange interpolation. The degree of the polynomial is limited to 3 or 4. Consider two vectors, the tangent to the function at the last point of the trajectory, and a line joining the last point of the trajectory to a candidate point match in the current frame. The cost associated with this match is related to the dot-product of the two vectors.

The three costs computed above are merged together for every candidate match and the Hopfield network is initialized to $v_{ij} = 1.0 - cost(i, j)$, where v_{ij} is the probability that the trajectory i till the previous frame matches point j in the current frame. The network is allowed to evolve and the hypotheses that survive are the right matches.

5 Eliminating wrong correspondences

Wrong correspondences that result from the network settling down to a local minimum are eliminated using a one-dimensional Hopfield network where each PE represents the probability of correctness of a correspondence. Each PE is connected to a processing element that represents a Voronoi neighbor correspondence that is most similar to it. The similarity measure is calculated as $similarity = 1.0 - cost = (1.0 - angcost)lengthcost + angcost$ where $lengthcost = 0.5[1.0 + \tanh(1.5(\Delta length - 2.0))]$ and $angcost = 0.5[1.0 + \tanh(0.15(\Delta angle - 20.0))]$. The network is simulated using the following equation for every PE

- $\frac{du_i}{dt} = -u_i + I_i + T_{i,j(i)}v_{j(i)}$
- $v_i = 0.5[1.0 + \tanh(\lambda u_i)]$

where $j(i)$ is the most similar neighbor to i , I_i is the bias input chosen to be 50 in all the experiments,

$T_{i,j(i)} = -100cost(i, j)$, and u_i and v_i are the input and output of the i^{th} PE. All those PEs that survive when the network stabilizes are retained. With this reduced set of correspondences, the entire process is repeated till no more correspondences are removed. This eliminates wrong correspondences.

6 Segmentation of trajectories

The underlying assumption made in the segmentation procedure proposed in this paper is that the motion between two frames is small due to dense sampling in time. The segmentation algorithm is outlined below. After the wrong correspondences are eliminated using the network discussed in Section 5, the Voronoi neighbors of every correspondence are determined. The lines joining the correspondences to their neighbors are called edges. These edges could be inter-object edges where the correspondences on either side belong to two different objects or within-the-object edges. For inter-object edges the similarity between the two correspondences on either end is low whereas for within-the-object edges it is high. We construct an energy function

$$E = \sum_{i=1}^{\#of\ edges} \lambda(1.0 - v_i)cost_i + \alpha v_i + \int_0^{v_i} g^{-1}(v)dv \quad (13)$$

that needs to be minimized to give the right segmentation. Here $cost_i$ is the cost reflecting the similarity of the correspondences on either end of the edge, v_i is the probability that the edge i is an inter-object edge and λ and α are constants weighing smoothness of motion versus discontinuities. This is minimized using the gradient descent approach according to the equations

- $\frac{du_i}{dt} = -\frac{\partial E}{\partial v_i} = -u_i + \lambda cost_i - \alpha$

where u_i and v_i are related as in the Hopfield network. All the inter-object edges are discarded. This gives the segmentation. This method fails when the motion between frames is very large or when the motion between neighboring objects is similar. The second case is inherently difficult to handle because if two nearby objects have similar motions, they act as a single object. In the first case, a better motion model must be found that fits unions of available segments.

7 Results

Experiments were conducted to evaluate the performance of the system on multiple, discontinuous motion image sequences and smooth motion real image sequences. The latter were taken from those made available at the 1991 IEEE workshop on visual motion.

Figure 1 shows the first frame of a sequence of 10 frames of a scene with 3 objects moving in different directions. Their motion varies from 4 to 10 pixels. 200 feature points corresponding to local intensity maxima and minima were automatically detected in every frame. The trajectories determined are shown in Figure

2. Figure 3 shows the result of segmenting the correspondences between the first two frames.

Figure 4 shows the first image of a sequence of 10 images of a scene with one object having a temporal discontinuity in its motion between the 5th and 6th frame. The object moves to the left for the first five frames and then moves towards the upper right hand corner of the image. Again around 200 points were automatically detected in each frame. Figure 5 shows the trajectories obtained.

Figure 6 shows the second frame of a sequence of images of a laboratory taken from a camera mounted on a PUMA robot arm that was rotating. This caused the entire scene to rotate around the optical axis of the camera. The motion of the features varied from 0 to 30 pixels. Figure 7 shows the correspondences obtained between the 2nd and 3rd frames of this sequence.

Figure 8 shows the 4th frame of a sequence of images of an outdoor scene taken by a camera mounted on a robot moving along the pathway seen in the image. The feature points had a motion of 15 – 20 pixels. Figure 9 shows the correspondences obtained between the 4th and 5th frames of this sequence.

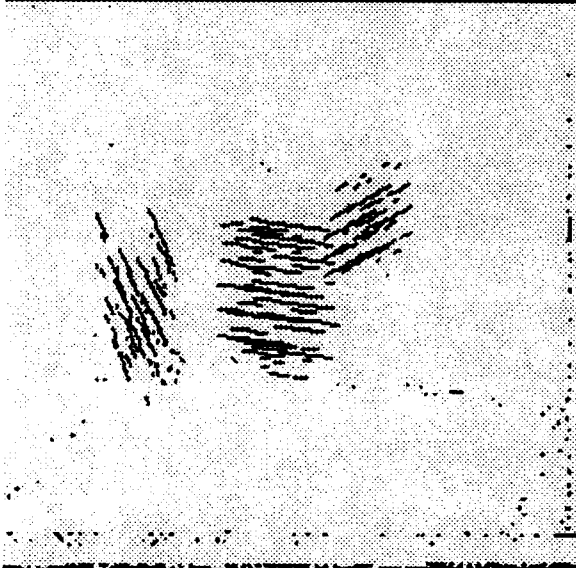
References

- [1] A. L. Yuille, "Energy functions for early vision and analog networks," *Biological Cybernetics*, vol. 61, pp. 115–123, 1989.
- [2] N. M. Grzywacz and A. L. Yuille, "Motion correspondence and analog networks," *Proceedings of the American Institute of Physics Conference on Neural Network Computing*, vol. 151, pp. 200–205, 1986.
- [3] Y. T. Zhou and R. Chellappa, "Neural network algorithms for motion stereo," in *Proceedings of the International Joint Conference on Neural Networks*, pp. II-251 – II-258, 1989.
- [4] Y. T. Zhou and R. Chellappa, "A network for motion perception," in *Proceedings of the International Joint Conference on Neural Networks*, pp. II-875 – II-884, 1990.
- [5] P. Y. Zhu, T. Kasvand, and A. Krzyzak, "Motion estimation based on point correspondence using neural network," in *Proceedings of the International Joint Conference on Neural Networks*, pp. II-869 – II-871, 1990.
- [6] J. J. Hopfield and D. W. Tank, "Neural computation in optimization problems," *Biological Cybernetics*, vol. 52, pp. 141–152, 1985.

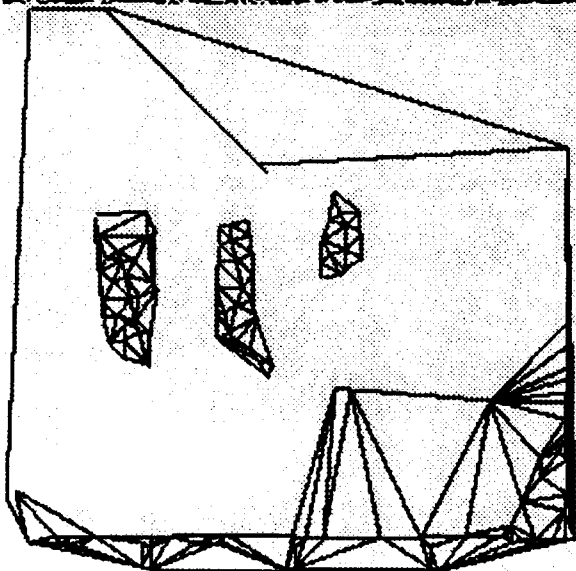
1.



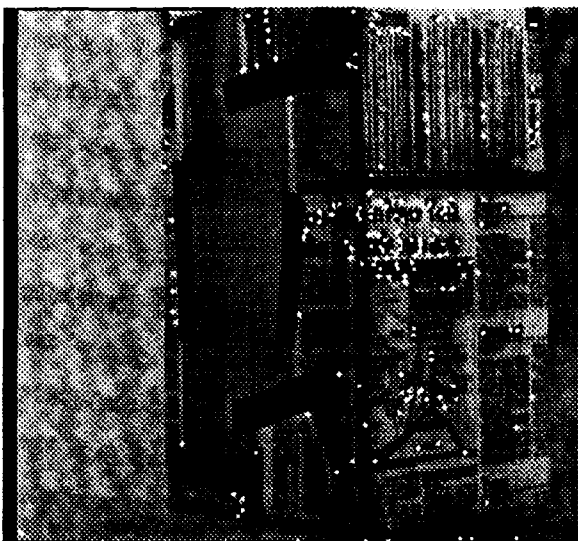
2.



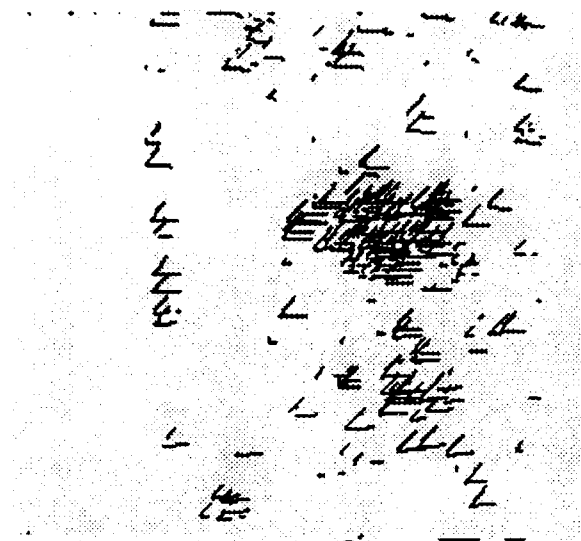
3.



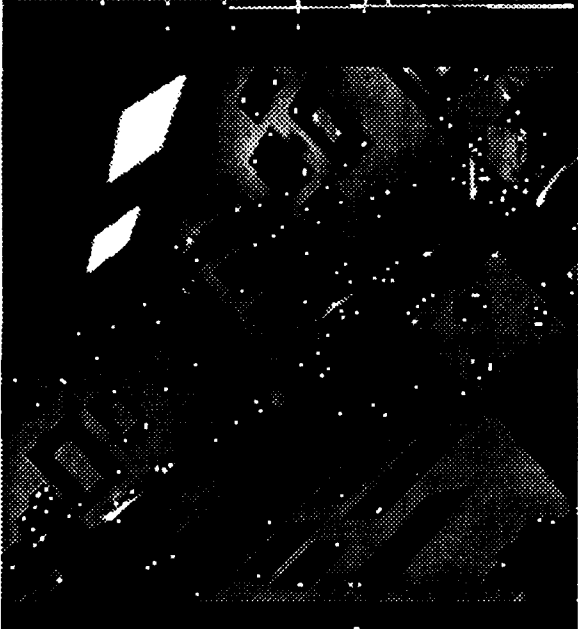
4.



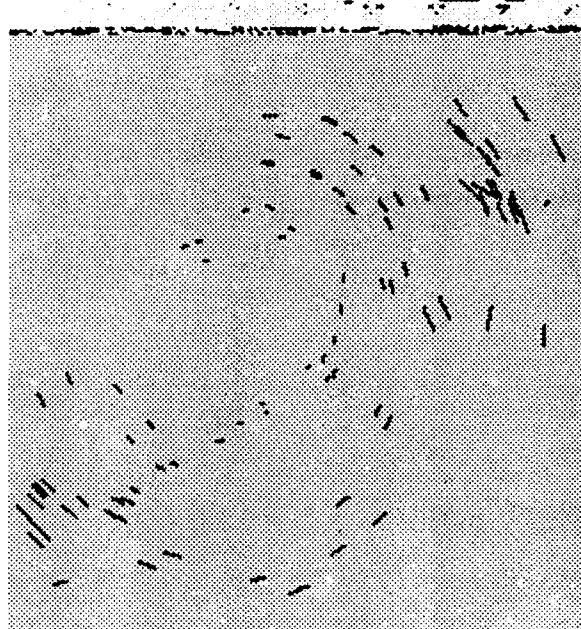
5.



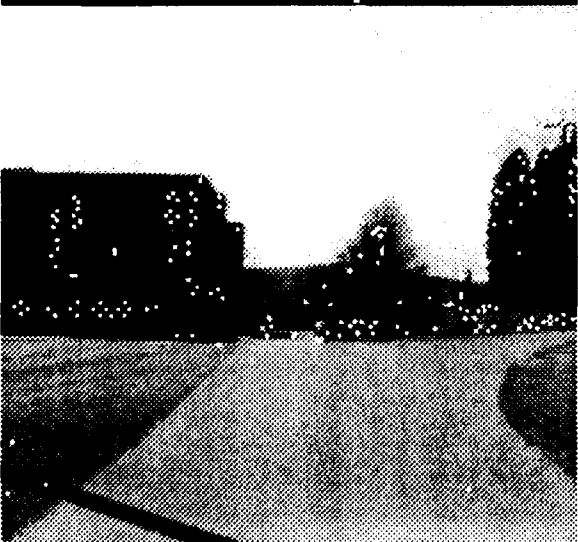
6.



7.



8.



9.



Motion constraint patterns

Cornelia Fermüller*

Computer Vision Laboratory,

Center for Automation Research

University of Maryland, College Park, MD 20742-3275

Abstract

The problem of egomotion recovery has been treated by using as input local image motion, with the published algorithms utilizing the geometric constraint relating 2-D local image motion (optical flow, correspondence, derivatives of the image flow) to 3-D motion and structure. Since it has proved very difficult to achieve accurate input (local image motion), a lot of effort has been devoted to the development of robust techniques. In this paper a new approach to the problem of egomotion estimation is taken, based on constraints of a global nature. It is proved that local normal flow measurements form global patterns in the image plane. The position of these patterns is related to the three dimensional motion parameters. By locating some of these patterns, which depend only on subsets of the motion parameters, through a simple search technique, the 3-D motion parameters can be found. The proposed algorithmic procedure is very robust, since it is not affected by small perturbations in the normal flow measurements. As a matter of fact, since only the sign of the normal flow measurement is employed, the direction of translation and the axis of rotation can be estimated with up to 100% error in the image measurements.

1. Introduction

The methodological theory of computational vision presented by Marr [15] has formed the basis for research in visual motion understanding. Vision is de-

scribed as a reconstruction process, that is, a problem of creating representations of increasing levels of abstraction, leading from 2-D images through the primal sketch and the $2\frac{1}{2}$ -D sketch to object-centered descriptions. Applied to visual motion perception this led to the computational theory known as "structure from motion" theory. The goal is to recover from dynamic imagery the 3-D motion parameters and the structure of the objects in view. The suggested strategy attempts to solve the problem in two stages. First, accurate image displacements between consecutive frames have to be computed, either in the form of point correspondences [7, 19] or as dense motion fields (optical flow fields) [4, 8, 11]. Then, in a second step, the 3-D motion and the structure are computed from the equations relating them to the 2-D image velocity [1, 5, 9, 13, 14, 17, 18]. This computational theory has been uncritically accepted, and as a result, most studies on visual motion perception are to be found at the algorithmic level of the problem, addressing either the estimation of image motion or the recovery of 3-D motion and structure from image motion.

The problem addressed in this paper is not the general "structure from motion" problem. We are concerned only with the estimation of 3-D motion independent of depth. For a monocular observer undergoing unrestricted rigid motion in the 3-D world, we compute the parameters describing this motion. Using the perspective transformation as our geometric imaging model, only five unknowns can be derived from 2D images, namely three rotational parameters and two parameters describing the direction of translation. In the literature this problem is known as "passive navigation".

In this paper an alternative approach to the problem of passive navigation is taken, which is different from existing methods, at both the computational and the algorithmic level.

First, we do not compute the exact 2-D image velocity. In general, the estimation of optical flow

*Permanent address: Department for Pattern Recognition and Image Processing, Institute for Automation, Technical University Vienna, Treitlstraße 3, A-1040 Vienna, Austria

is an ill-posed problem and additional assumptions must be made in order to estimate it. If these assumptions hold, as in the case of some model-based approaches [6, 12], for special purpose applications, optical flow can be computed and as a result 3-D motion can be derived. In the general case, however, any algorithm will produce imperfect output (erroneous output, if the assumptions do not hold). Therefore, we use as a representation for image motion the so-called normal flow. As the only available constraint on the flow (u, v) of the time varying image $I(x, y, t)$ we consider the motion constraint equation [11] $I_x u + I_y v + I_t = 0$, where the subscripts denote partial differentiation. This constraint means that we can only compute the projection of the flow on the gradient direction $((I_x, I_y) \cdot (u, v) = -I_t)$.

Recovering 3-D motion from noisy flow fields has turned out to be a problem of extreme sensitivity, with researchers reporting very large errors in the motion parameter estimates under small perturbations in the input. Even optimal algorithms [17] perform quite poorly in the presence of moderate noise. Although a formal proof is still lacking, it has been argued [2] that the estimation of 3-D motion from image motion is itself ill-posed, because it does not continuously depend on the input.¹ Thus, in order to estimate 3-D motion in a robust way, we have to consider the fact that no flow measurement (neither optical flow nor normal flow) can be perfect. In this paper new global constraints of a geometric nature, which relate 3-D motion to 2-D image measurements (normal flow), are introduced.

In our approach, we first compute the rotation axis and the direction of translation. Motion rigidity introduces a number of constraints on the normal flow values. These constraints take the form of patterns in the image plane. In other words, for given positions of the translational and rotational axes, the normal flow values form certain global patterns. Our algorithmic procedure searches for these patterns. It uses data from different parts of the image plane and considers only the sign of the normal flow. The method for deriving the direction of translation and the rotation axis is of a robust and global character and thus can handle a considerable amount of error in the input. After having found the axis of rotation and the direction of translation further constraints are considered, and the complete set of motion parameters is obtained.

Methods for estimating 3-D motion from only the normal flow field without going through the intermediate stage of computing optical flow have previously appeared in [3, 10, 16]. In [3] the case of purely

rotational motion was studied, and linear equations relating the rotation parameters to the normal flow were derived. A similar result was reported by Horn and Weldon [10], who presented several methods for solving the problem of motion and structure computation not only in the purely rotational case, but also for pure translation, for known rotation, and for known structure. The constraint of positive depth was used by Negahdaripour [16] to estimate the focus of expansion for purely translational motion and in [20] translation and rotation were estimated for an observer rotating around the direction of translation.

2. Geometric constraints

For a monocular observer undergoing unrestricted rigid motion in the 3-D world we compute the parameters describing this motion. If the coordinate system is fixed to the observer with the center being the nodal point of the camera and f the focal length, then the equations relating the velocity (u, v) of an image point to the 3-D velocity $((U, V, W)$ translational and (α, β, γ) rotational) and the depth Z of the corresponding scene point are [13]

$$\begin{aligned} u &= \frac{(-Uf + xW)}{Z} + \alpha \frac{xy}{f} - \beta \left(\frac{x^2}{f} + f \right) + \gamma y \\ v &= \frac{(-Vf + yW)}{Z} + \alpha \left(\frac{y^2}{f} + f \right) - \beta \frac{xy}{f} - \gamma x \end{aligned} \quad (1)$$

The number of motion parameters a monocular observer is able to compute under perspective projection is limited to five: the three rotational parameters and the direction of translation. We therefore introduce coordinates for the direction of translation, $(x_0, y_0) = (Uf/W, Vf/W)$ and rewrite the righthand side of equation (1) as a sum of translational and rotational components:

$$u = u_{\text{trans}} + u_{\text{rot}} = \quad (2)$$

$$= (-x_0 + xf) \frac{W}{Z} + \alpha \frac{xy}{f} - \beta \left(\frac{x^2}{f} + f \right) + \gamma y$$

$$v = v_{\text{trans}} + v_{\text{rot}} = \quad (3)$$

$$= (-y_0 + yf) \frac{W}{Z} + \alpha \left(\frac{y^2}{f} + f \right) - \beta \frac{xy}{f} - \gamma x \quad (4)$$

Since we can only compute normal flow, the projection of flow on the gradient direction (n_x, n_y) (unit vector), only one constraint can be derived at every point. The value u_n of the normal flow vector along the gradient direction is given by

$$u_n = un_x + vn_y, \text{ or}$$

¹A problem is ill-posed if its solution does not exist, is not unique, or does not continuously depend on the input.

$$u_n = ((-x_0 + xf)\frac{W}{Z} + \alpha\frac{xy}{f} - \beta(\frac{x^2}{f} + f) + \gamma y)n_x \\ + ((-y_0 + yf)\frac{W}{Z} + \alpha(\frac{y^2}{f} + f) - \beta\frac{xy}{f} - \gamma x)n_y \quad (5)$$

The above equation demonstrates the difficulties of motion computation using normal flow. A monocular observer not being able to measure depth is confronted with a motion field of five unknown motion parameters and one scaled depth component (W/Z) at every point. Since there is only one constraint for a single point and since we do not want to make assumptions about depth, there is no straightforward way to compute the motion parameters analytically.

2.1. Motion field interpretation

A motion field is composed of a translational and a rotational component. Only the first of these is dependent on the distance from the observer. This suggests the idea of searching for a way of determining the motion components by disregarding the depth components. The motion under consideration is rigid. Every point in 3-D moves relative to the observer along a constrained trajectory. The rigidity constraint also imposes restrictions on the motion field in the image plane and these restrictions are reflected in the normal field as well. This is the motivation for investigating the geometrical properties inherent in the normal flow field. The motion estimation problem then amounts to resolving the normal flow field into its rotational and translational components.

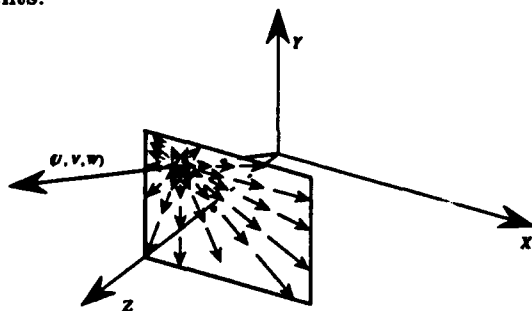


Figure 1: Translational motion viewed under perspective projection: the observer is approaching the scene.

If the observer undergoes only translational motion, all points in the 3-D scene move along parallel lines. Translational motion viewed under perspective results in a motion field in the image plane, in which every point moves along a line that passes through a single vanishing point. This point is the intersection of the image plane and the line parallel to the translation direction that passes through the nodal

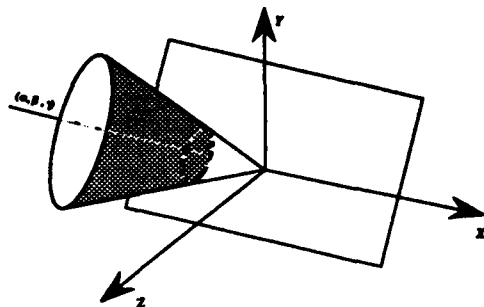


Figure 2: The intersection of the image plane with a cone (defined by the circular path in 3-D and the rotation axis) defines the projection of rotational motion on the image plane.

point. Its image coordinates are $x = Uf/W$ and $y = Vf/W$; the flow there has value zero.

In cases where the sensor is approaching the scene all the image motion vectors emanate from the vanishing point, which is then called Focus of Expansion (FOE) (Figure 1). Otherwise they all point into it, in which case we speak of the Focus of Contraction (FOC). The direction of every vector is determined by the location of the vanishing point, and the length of each vector is dependent on the 3-D position of the corresponding scene point. The vanishing point also constrains the direction of the normal flow vector at every point; it can only be in the half plane containing the optical flow vector at that point.

In cases of purely rotational motion every point in 3-D moves along a circle in a plane perpendicular to the axis of rotation. The perspective image of this circular path is the intersection of the image plane with the cone which the circle defines together with the rotation axis (see Figure 2). Depending on the relation between the opening angle of the cone for a specific image point and the angle the image plane forms with the rotation axis, the field lines of the rotational vector field (i.e the lines which have the property that at each point the rotational flow is tangential to them) form second order curves of different types: ellipses, hyperbolas, parabolas, or even circles when the rotation axis and the optical axis coincide. The conic sections generated by a rotational motion are defined by the axis of rotation. A rotation axis, given by the two parameters $(\frac{\alpha}{\gamma})$ and $(\frac{\beta}{\gamma})$, defines a family $M(\frac{\alpha}{\gamma}, \frac{\beta}{\gamma}; x, y)$ of conic sections:

$$M(\frac{\alpha}{\gamma}, \frac{\beta}{\gamma}; x, y) = ((\frac{\alpha}{\gamma})^2 x^2 + 2xy\frac{\alpha}{\gamma}\frac{\beta}{\gamma} + y^2(\frac{\beta}{\gamma})^2 \\ + 2xf\frac{\alpha}{\gamma} + 2yf\frac{\beta}{\gamma} + f^2)/(x^2 + y^2 + f^2) = C \quad (6) \\ \text{with } C \text{ in } [0 \dots (1 + (\frac{\alpha}{\gamma})^2 + (\frac{\beta}{\gamma})^2)]$$

3. Properties of selected vectors

In this section geometrical properties of normal flow vectors in selected directions are investigated. To be more precise, we study the sign of the normal flow in certain directions and the locations of normal flow vectors of the same sign. Vectors which are perpendicular to rotational vector field lines and vectors perpendicular to lines emanating from a point are considered. For these vectors we find that the normal flow values in the image are separated into regions in which they have different signs by a second order curve and a straight line.

The normal flow vector \vec{u}_n is the projection of the optical flow vector \vec{u} on the gradient direction and the value of the normal flow is therefore defined by the scalar product of the optical flow vector and the unit vector (n_x, n_y) in the gradient direction. The flow vector can be decomposed into its translational and rotational components and the right hand side of equation (5) can be written as a sum of scalar products:

$$u_n = \frac{W}{Z}((-x_0 + xf), (-y_0 + yf))(n_x, n_y) + ((\alpha \frac{x^2}{f} - \beta(\frac{x^2}{f} + f) + \gamma y), (\alpha(\frac{y^2}{f} + f) - \beta \frac{x^2}{f} - \gamma x)(n_x, n_y) \quad (7)$$

Our goal is to achieve a separation between translation and rotation. Therefore we classify the normal flow vectors according to their direction by defining two classes which are motivated by the concepts of the rotation axis and the FOE.

Any possible axis given by an orientation vector (A, B, C) , where $A^2 + B^2 + C^2 = 1$, defines an infinite class of cones with axis (A, B, C) and apex at the origin. The image plane gives rise to a set of conic sections, hereafter called vector field lines, or field lines of the axis (A, B, C) , or just (A, B, C) field lines. It is worth noting that the (A, B, C) field lines are the lines along which the image points would move if the observer rotated around axis (A, B, C) . Normal flow vectors are combined into a single class if they are perpendicular to the vector field lines of the same axis (A, B, C) . At a point (x, y) the orientation perpendicular to the vector field lines (A, B, C) is given by a vector $\vec{N} = (N_x, N_y)$:

$$(N_x, N_y) = ((-A(y^2 + f^2) + Bxy + Cxf), (Axy - B(x^2 + f^2) + Cyf))$$

and the unit vector $\vec{n} = (n_x, n_y)$ denoting the gradient is thus $\vec{n} = \frac{\vec{N}}{\|\vec{N}\|}$. We call the vectors of the class corresponding to the axis (A, B, C) the coaxis vectors (A, B, C) . In order to establish conventions about the vectors' orientations, a vector will be said

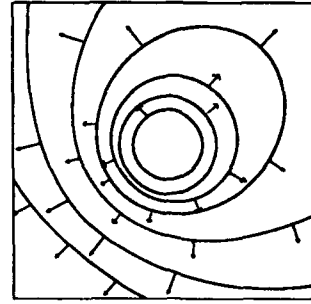


Figure 3: Field lines corresponding to an axis (A, B, C) and positive coaxis vectors (A, B, C) .

to be of positive orientation if it is pointing in direction $\vec{n} = (n_x, n_y)$, whereas, if it is pointing in direction $(-n_x, -n_y)$, its orientation will be said to be negative (see Figure 3).

Next we evaluate the translational components of the normal flow vectors in the chosen direction. The value t_n of any translational vector component at point (x, y) in direction (n_x, n_y) is given by

$$t_n = ((x - x_0, y - y_0) \frac{W}{Z}) \vec{n}$$

Since $\frac{W}{Z}$ is positive when the observer is approaching the scene, a classification into positive and negative values independent of the distance from the image plane is possible. The translational components of the coaxis vectors (A, B, C) are separated by a second order curve $h(A, B, C, x_0, y_0; x, y)$ given by

$$h(A, B, C, x_0, y_0; x, y) = x^2(Cf + By_0) + y^2(Cf + Ax_0) - xy(Ay_0 + Bx_0) - xf(Af + Cx_0) - yf(Bf + Cy_0) + f^2(Ax_0 + By_0) = 0. \quad (8)$$

When $h(x, y) > 0$ the translational normal flow values are positive; when $h(x, y) < 0$ they are negative; and when $h(x, y) = 0$ they have value zero. For any selected class of coaxis vectors there exists a curve h which is uniquely defined by the two coordinates x_0, y_0 of the FOE; furthermore it is linear in x_0 and y_0 (see Figure 4a).

The rotational components of the flow vectors are defined only by the three rotational parameters α, β and γ . Along the positive direction of the coaxis vectors the value r_n of the rotational components is

$$r_n = ((\alpha \frac{x^2}{f} - \beta(\frac{x^2}{f} + f) + \gamma y), (\alpha(\frac{y^2}{f} + f) - \beta \frac{x^2}{f} - \gamma x)(n_x, n_y))$$

The coaxis vectors (A, B, C) and the rotational flow vectors form a right angle for all points on a straight

line. Thus, considering only the sign of the rotational component along the coaxis vectors (A, B, C) , the image plane is separated by a straight line $g(A, B, C, \alpha, \beta, \gamma)$ into two halves containing values of opposite sign, where

$$g(A, B, C, \alpha, \beta, \gamma, x, y) = y(\alpha C - \gamma A) - x(\beta C - \gamma B) + \beta A f - \alpha B f = 0 \quad (9)$$

Again the sign of $g(x, y)$ at a point (x, y) determines the sign of the coaxis vectors (A, B, C) . The straight line is defined by only two parameters which characterize the axis of rotation, namely $\frac{\alpha}{\gamma}$ and $\frac{\beta}{\gamma}$ (see Figure 4b).

In order to investigate the constraints for general motion the geometrical relations due to rotation and due to translation have to be combined. A second order curve separating the plane into positive and negative values and a line separating the plane into two halfplanes of opposite sign intersect. This splits the plane into areas of only positive coaxis vectors, areas of only negative vectors, and areas in which the rotational and translational flow have opposite signs. In these last areas, no information is derivable without making depth assumptions (Figure 4c).

We thus obtain the following geometrical result for the case of general motion. Any class of coaxis vectors (A, B, C) is separated by a rigid motion into two groups. The FOE (x_0, y_0) and the rotation axis $(\frac{\alpha}{\gamma}, \frac{\beta}{\gamma})$ geometrically define two areas in the plane, one containing positive and one containing negative values. We call this structure on the coaxis vectors the coaxis pattern. It depends on the four parameters $x_0, y_0, \frac{\alpha}{\gamma}$ and $\frac{\beta}{\gamma}$.

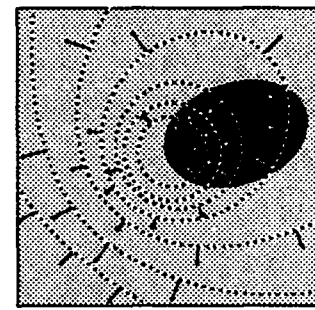
For the second kind of classification of the normal flow vectors, namely the one defined as "perpendicular to the lines emanating from a defined point" (see Figure 5), similar patterns are obtained. In this case, the rotational components are separated by a second order curve into positive and negative values and the translational components are separated by a straight line. We call the vectors perpendicular to straight lines passing through a point (r, s) the copoint vectors (r, s) .²

At point (x, y) a copoint vector \vec{n} of unit length in the positive direction is defined as

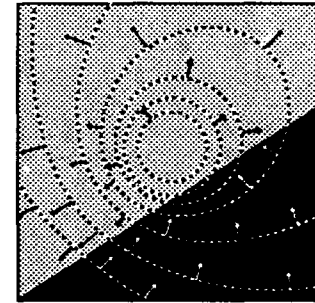
$$\vec{n} = \frac{(-y + s, x - r)}{\sqrt{(x - r)^2 + (y - s)^2}}$$

The functions which define the curves are given as follows: The straight line $k(r, s, x_0, y_0, x, y)$ separating

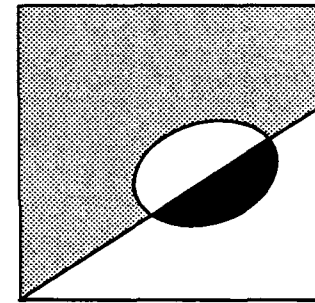
²The copoint and coaxis vectors are dual to each other.



(a)



(b)



(c)

Figure 4: (a) The coaxis vectors (A, B, C) due to translation are negative if they lie within a second order curve defined by the FOE, and are positive at all other locations. (b) The coaxis vectors due to rotation separate the image plane into a halfplane of positive values and a halfplane of negative values. (c) A general rigid motion defines an area of positive coaxis vectors and an area of negative coaxis vectors. The rest of the image plane is not considered.

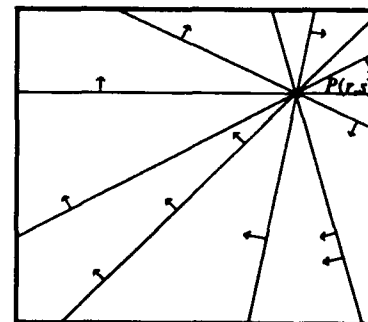


Figure 5: Positive copoint vectors (r, s) .

the translational components is

$$k(r, s, x_0, y_0; x, y) = y(x_0 - r) - x(y_0 - s) - x_0 s + y_0 r = 0 \quad (10)$$

and the second order curve $l(r, s, \alpha, \beta, \gamma; x, y)$ separating the rotational components is (like $h(A, B, C, x_0, y_0, x, y)$) defined as

$$l(r, s, \alpha, \beta, \gamma, x, y) = -x^2(\beta s + \gamma f) - y^2(\alpha r + \gamma f) + xy(\alpha s + \beta r) + xf(\alpha f + \gamma r) + yf(\beta f + \gamma s) - f^2(\alpha r - \beta s) = 0 \quad (11)$$

The superposition of translational and rotational values again defines patterns in the plane which consist of a negative and a positive area. These patterns, called copoint patterns, are defined by the same four parameters which characterize the coaxis patterns.

4. Search for motion patterns

Utilizing the geometrical constraints developed in the last section, motion estimation for a rigid moving observer will now be addressed through a search technique. The strategy involves checking constraints that a certain solution would impose on the normal flow field and in this way discarding impossible solutions. The search is performed in three steps, where at each step the constraints become more restrictive; hence the number of possible solutions computed at each step decreases. First a set S_1 of possible solutions for the FOE and axis of rotation is estimated by fitting a small number of patterns to the normal flow field. Two techniques, which use different patterns defined on certain coaxis vectors, are proposed for solving this task. Both fitting processes use the input in a qualitative way, since only the sign of the normal flow is employed. In the second step the third rotational parameter is computed, and the space of solutions is further narrowed to a set S_2 . This is performed by using normal flow vectors that do not contain translation (certain copoint vectors) and thus approximating the remaining rotational parameter from the given rotational vectors. Finally, in the last step all impossible solutions are discarded by checking the validity of the motion parameters at every point, which results in a set S_3 as output.

4.1. First step: Pattern fitting

The direction of translation and the axis of rotation define patterns on subsets of the normal flow vectors. In the general case these patterns are described by four independent variables and searching for the solution would mean searching in a four-dimensional parameter space. By concentrating, in an initial search, only on a small number of normal flow vectors, we

show how to tackle the problem. Clearly, such a restricted use of data will generally not result in a unique solution, but it allows us to either reduce the dimensionality of the problem (algorithm 1) or to employ motion vectors from all parts of the image plane (algorithm 2).

Algorithm 1.1 : α -, β -, and γ -pattern fitting

One way to look at the optical flow vector is to imagine it as a sum of five vectors, each being due to only one of the motion parameters (either one of the two translational or one of the three rotational components). Consequently the value of the normal flow vector at a point is computed as the sum of the five scalar products of these vectors and the unit vector in the gradient direction. The scalar product of two vectors is zero if the vectors are perpendicular to each other. Thus, by selecting normal flow vectors in particular directions, one or more of the motion components vanish.

The coaxis vectors which are dependent on only two of the three rotational parameters correspond to one of the three coordinate axes. These normal vectors and their patterns have special properties.

The coaxis vectors (A, B, C) when the orientation vector (A, B, C) is the Z axis are perpendicular to circles whose center is the origin of the image plane, and we call them γ -vectors. Similarly, when (A, B, C) is the X or Y axis, the (A, B, C) coaxis vectors are called α -vectors and β -vectors and the corresponding field lines are hyperbolas whose major axes are the image plane's x - and y -axes, respectively. Figure 6 depicts these sets of vector field lines and the corresponding γ -, α - and β -vectors in positive orientation.

The values of the α -, β -, and γ -vectors due to rotation only can be described by a one-parameter function. Thus the dimensionality of the corresponding patterns is also reduced by one and the search for these patterns can be limited to a three-dimensional parameter-space. This becomes clear by substituting into equation (9) for the triple (A, B, C) the orientation vectors of the coordinate axes $((1, 0, 0), (0, 1, 0)$ and $(0, 0, 1))$. The rotational components of the γ -vectors are separated by a line passing through the center, which has equation $y = \frac{\beta}{\alpha} x$. For the rotational components of the α -vectors the line is parallel to the x -axis and is defined by the equation $x = \frac{\alpha}{\gamma} f$. The β -vectors are separated by a line parallel to the y -axis defined as $x = \frac{\alpha}{\gamma} f$.

The second order curves separating the translational components of the α -, β -, and γ -vectors are obtained from equation (8). For the γ -vectors the curve reduces to a circle, which has the FOE and the image

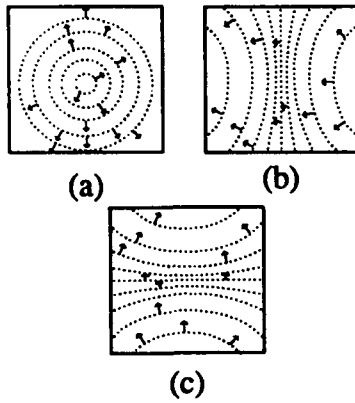


Figure 6: If the (A, B, C) axis is the Z -, X -, or Y -axis, the corresponding vector field lines are circles with center O (a), or hyperbolas whose axes coincide with the coordinate axes of the image plane (b and c). Normal flow vectors perpendicular to these field lines are called γ -, α -, and β -vectors.

center as two diametrically opposite points. Equation (8) reduces to

$$h(0, 0, 1, x_0, y_0; x, y) = f(x - \frac{x_0}{2})^2 + f(y - \frac{y_0}{2})^2 - (\frac{x_0}{2})^2 + (\frac{y_0}{2})^2 = 0$$

The curves separating the α - and β -vectors become hyperbolas of the form

$$h(1, 0, 0, x_0, y_0; x, y) = y^2 x_0 - xy y_0 - x f^2 + f^2 x_0 = 0$$

and

$$h(0, 1, 0, x_0, y_0; x, y) = x^2 y_0 - xy x_0 - y f^2 + f^2 y_0 = 0$$

Figure 7 shows the α -, β -, and γ -vectors for a general rigid motion.

The algorithm which computes the FOE and the axis of rotation from a given normal flow field by using only the α -, β - and γ -vectors works as follows. With each subset of normal flow vectors is associated a three-dimensional parameter space that spans the possible locations of the FOE and of a line defined by the quotient of two of the three rotational parameters. A search in the three-dimensional subspaces is accomplished by checking the patterns which the subspaces' parameter triples define for selected values of the normal flow field. The α -patterns are fitted to the α -vectors, which provides possible solutions for the coordinates of the FOE: x_0, y_0 , and the quotient $\frac{\beta}{\alpha}$. Similarly, the fitting of the β - or γ -patterns results in solutions for x_0, y_0 , and $\frac{\beta}{\gamma}$ or $\frac{\alpha}{\gamma}$. The objective is to find the four parameters defining the directions of the translational and rotational axes which give rise to

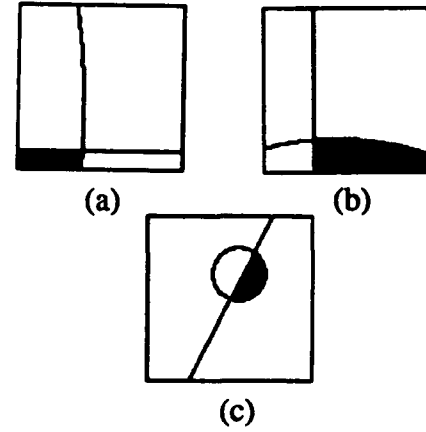


Figure 7: α -, β -, and γ - patterns for a general rigid motion.

three successfully fitted patterns. Therefore the three subspaces' patterns are combined and the parameter quadruples which define possible solutions are determined. Since only subsets of the normal flow values are considered in the fitting process, the fitting alone does not uniquely define the motion, but just constitutes a necessary condition. Usually there will be a number of parameter quadruples $\{x_0, y_0, \alpha/\gamma, \beta/\gamma\}$ that are selected as candidate solutions through pattern fitting.

The range of values for the coordinates of the FOE and for $\frac{\beta}{\gamma}$ and $\frac{\alpha}{\gamma}$ is $[-\infty, +\infty]$. To cope with all possible cases a coordinate transformation on the sphere is performed, in which case the coordinates are expressed by two angles.

Algorithm 1.2: Search for the rotation axis

For any rigid motion there exists one class of coaxial vectors which does not contain any rotational components. This set is defined by the actual rotation axis $\frac{A}{C} = \frac{\alpha}{\gamma}$ and $\frac{B}{C} = \frac{\beta}{\gamma}$. The coaxial vectors of this kind are due only to translation and the pattern of these vectors is solely defined by the two-parameter second order curve $h(\alpha, \beta, \gamma, x_0, y_0; x, y)$. There is only one curve separating the positive from the negative values and thus the pattern is defined on the whole image plane. Since $h(\alpha, \beta, \gamma, x_0, y_0; x, y)$ is linear in x_0 and y_0 the problem of finding the FOE from the normal vectors due to rotation reduces to estimating the linear discriminant function separating two classes (labeled positive and negative) of values.

The pattern is due only to two parameters. In order to find the axis of rotation a search in the two-dimensional parameter space of $\frac{\alpha}{\gamma}$ and $\frac{\beta}{\gamma}$ is performed. For every possible rotation axis the data is checked for linear discrimination. If a second order curve can be found that separates the positive from

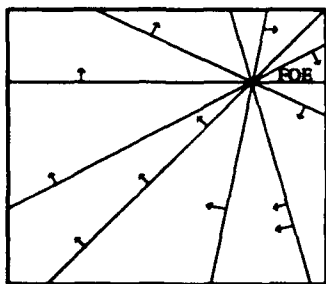


Figure 8: Normal flow vectors perpendicular to lines passing through the FOE are due only to rotation.

the negative values the quadruple $(x_0, y_0, \frac{a}{\gamma}, \frac{b}{\gamma})$ will be added to the set of possible solutions.

Concerning the computational aspect of solving the discrimination problem, different algorithms from the pattern recognition literature can be applied. For example, the Ho-Kashyap algorithm decides whether a data set is linearly discriminable and will also find the best discrimination.

4.2. Second step: Detranslation

Proper selection of normal flow vectors also enables the elimination of the normal flow's translational components. This can be achieved by choosing as normal flow vectors the copoint vectors defined by the locus of the FOE. With the location of the FOE the directions of the translational motion components are defined. The optical flow vectors lie on lines passing through the FOE. The normal flow vectors perpendicular to these lines (the copoint vectors (r, s) , where $r = x_0$ and $s = y_0$), do not contain translational, but only rotational components. This can be seen from equation (5). If the selected gradient direction at a point (x, y) is $((y_0 - y), (-x_0 + x))$ the scalar product of the translational motion component and a vector in the gradient direction is zero. This method of eliminating the translational component, referred to below as "detranslation", is applied to compute the third rotational component and to further reduce the possible number of solutions.

For each of the possible solutions computed in the second module, the normal flow vectors perpendicular to the lines passing through the FOE have to be tested to see if they are only due to rotation (see Figure 8). This results in solving an overdetermined system of linear equations. Since two of the rotational parameters are already computed, there is only one unknown, the value γ . Every point supplies

an equation of the form

$$\gamma = \frac{u_n(\frac{a}{\gamma}(\frac{x}{f}n_x + (\frac{y}{f} + f)n_y) - \frac{b}{\gamma}((\frac{x}{f} + f)n_x + \frac{y}{f}n_y) + (yn_x - xn_y))}{(12)}$$

Since the chosen normal flow vectors are due only to rotation, the solution to the overdetermined system gives the γ value. In a practical application a threshold has to be chosen to discriminate between possible and impossible solutions. The value of the residual is used to confirm the presumption that the selected normal flow values are purely rotational. Usually "detranslation" will not result in only one solution, but will provide a set S_2 of possible parameter quintuples.

4.3. Third step: Derotation

The modules described so far considered only subsets of the normal flow vectors. Clearly, after having found possible solutions for the FOE and the axis of rotation, we can test every candidate solution for its correctness on any class of coaxial vectors. Since the quadruple $(x_0, y_0, \frac{a}{\gamma}, \frac{b}{\gamma})$ defines a pattern on every class of coaxial vectors, we just have to test for the existence of this pattern. However, a pattern in the general case is defined only on parts of the image plane. Thus even by testing every possible class of coaxial vectors not every normal flow vector will be tested.

In order to eliminate all motion parameters which are in contradiction to the given normal flow field, every normal flow vector has to be checked. This check is performed by a "derotation" technique. With every parameter quintuple computed in the second step a possible FOE and a rotation are defined. The three rotational parameters are used to derotate the normal flow vectors by subtracting the rotational component (u_{rot}, v_{rot}) . At every point the flow vector (u_{der}, v_{der}) is computed:

$$\begin{aligned} u_{der} &= u_n n_x - u_{rot} n_x \\ v_{der} &= u_n n_y - v_{rot} n_y \end{aligned} \quad (13)$$

If the parameter quintuple defines the correct solution, the remaining normal flow is purely translational. Thus, it has to have the property of an emanating motion field. Since the direction of optical flow for a given FOE is known, the possible directions of the normal flow vectors can be determined. The normal flow vector at every point must lie in a half plane (see Figure 9). The technique checks all points for this property and eliminates solutions that cannot give rise to the given normal flow field.

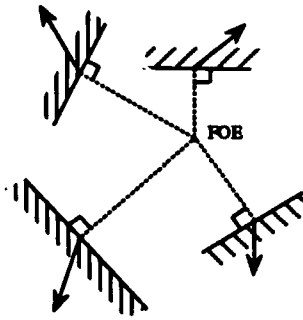


Figure 9: Normal flow vectors due to translation are constrained to lie in halfplanes.

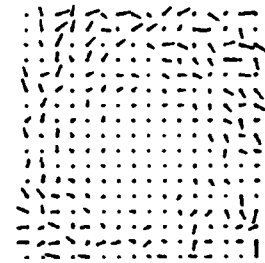
4.4. The complete technique

In this section we give a summary of the complete technique in the form of a block diagram. The computation of an observer's egomotion is performed in three steps, where for the first step two alternative modules can be chosen. The sets of candidate solutions which are determined in the four modules are called S_1 , S_2 , S_3 . To denote single solutions or single parameters, subscripts are used: $S_{1,i}$, $S_{2,i}$, $x_{0,i}$, $y_{0,i}$, etc. The input to the algorithm is a normal flow field and the outputs are all possible solutions for the direction of translation and the rotation which can give rise to this normal flow field.

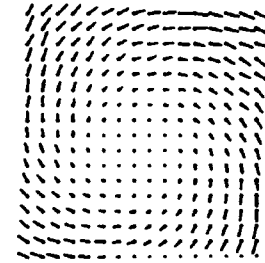
The algorithm determines the complete set of solutions. If for a given normal flow field the algorithm finds more than one solution, then from the normal flow field alone the 3-D motion cannot be determined uniquely. In this case one may use matching of prominent features to eliminate the incorrect motion parameters.

5. Experiments

Several experiments have been performed on synthetic and real data. For different 3-D motion parameters normal flow fields were generated, where the depth value within an interval and the gradient direction were chosen randomly. In all experiments on noiseless data the correct solution was found as the best one. Figure 10 shows the optical flow field and the normal flow field for one of the generated data sets. The image size is 100×100 , the focal length is 150, the image coordinates of the FOE are $(-5, +30)$ and the relationship between the rotational components is $\alpha : \beta : \gamma = 10 : 11 : 150$. In Figure 11 the fitting of the circle and the hyperbolas to the α -, β -, and γ -vectors and the coaxis pattern $(\frac{\alpha}{\gamma}, \frac{\beta}{\gamma})$ is displayed. Points with positive normal flow values are displayed in a light gray level and points with negative values are dark. In the two modules of the first



Normal flow field



Optical flow field

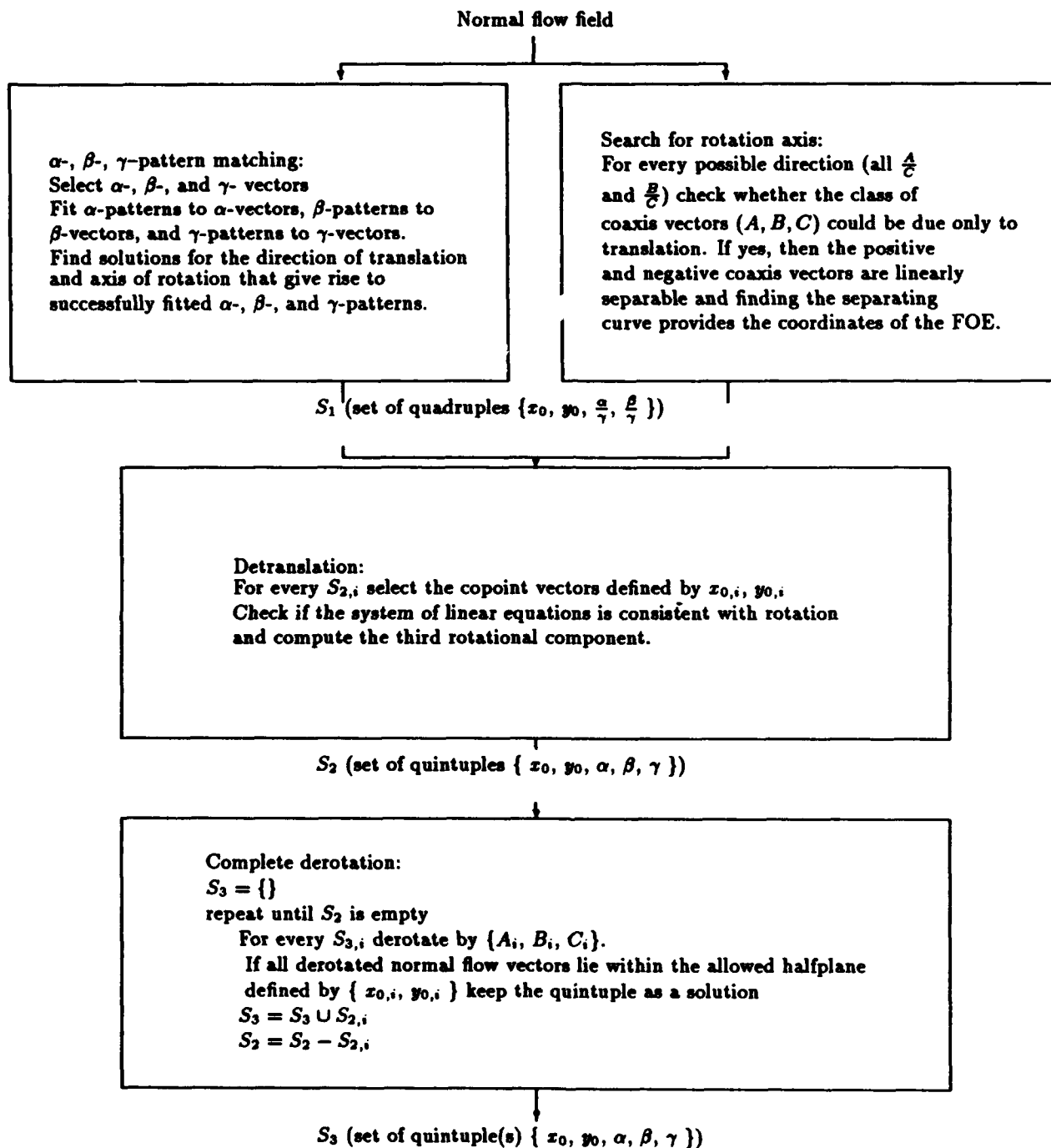
Figure 10: Flow fields of synthetic data.

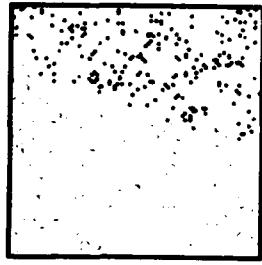
step no quantitative use of values is made, since only the sign of the normal flow is considered. This limited use of data makes the module very robust, and the correct solution for the axes of translation and rotation will be found even in the presence of high amounts of noise (up to 100%).

The NASA-Ames sequence³ was used as a real data set. In this sequence the camera undergoes only translational motion; we added different amounts of rotation. For all points at which the translational motion can be found, the rotational normal flow is computed and the new position of each pixel is evaluated. The "rotated" image is then generated by computing the new greyvalues through bilinear interpolation. The images were convolved with a Gaussian of kernel size 5×5 and standard deviation $\sigma = 1.4$. The normal flow was computed by using 3×3 large Sobel operators to estimate the spatial derivatives in the x - and y -directions and by subtracting the 3×3 box-filtered values of consecutive images to estimate the temporal derivatives.

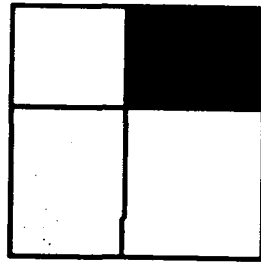
When adding rotational normal flow of magnitude on the order of a third to three times the amount of translational flow, the exact solution was always found among the best fitted parameter sets. In Figure 12 the computed normal flow vectors and the fitting of the α -, β -, and γ -vectors for one of the "rotated" images is shown. Areas of negative normal flow vectors are marked by horizontal lines and areas of positive values by vertical lines. The ground

³This is a calibrated motion sequence made public for the Workshop on Visual Motion, 1991.





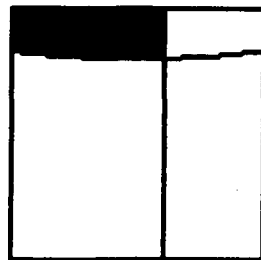
(a)



(d)



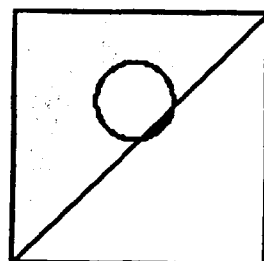
(b)



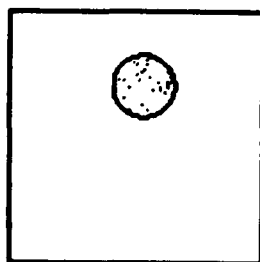
(e)



(c)

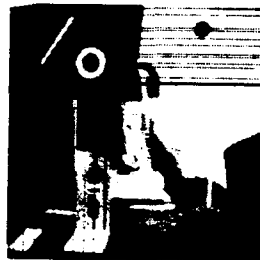
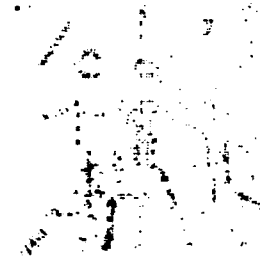


(f)

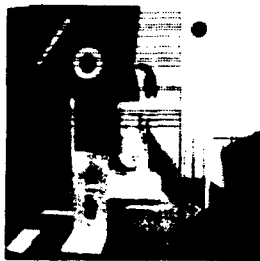


(g)

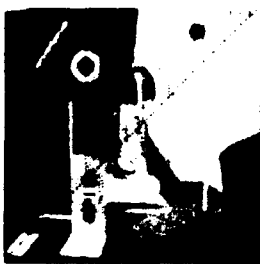
Figure 11: (a),(b),(c): Positive and negative α -, β -, and γ -vectors of synthetic data. (d),(e),(f): Fitting of α -, β -, and γ -patterns. (g): Separation of coaxial pattern ($\frac{\alpha}{\gamma}$, $\frac{\beta}{\gamma}$).



α



β



γ

Figure 12: Natural scene: Normal flow field and fitting of α -, β -, and γ -vectors.

truth for the FOE is $(-5, -8)$, the focal length is 599 pixels and the rotation between two image frames is $\alpha = 0.0006$, $\beta = 0.0006$ and $\gamma = 0.004$. The algorithm computed the solution exactly.

6. Conclusions

We have presented geometric constraints on normal flow fields due to rigid motion, which take the form of patterns in the image plane. These constraints were exploited to solve the problem of computing the 3-D motion of an observer relative to the scene in a robust way. We claim robustness, because for the estimation of the translational and rotational axes only the signs of the normal flow vectors are used; and the technique is global, because values in all parts of the image are considered. The algorithmic procedure constitutes a search technique in a parameter space, where appropriate selection of normal flow values is used in different ways to reduce the dimensionality of the motion estimation problem. In order to compute the axes of translation and rotation two different solutions were proposed. Either three different subsets of the vector field are searched for patterns defined by only three of the five parameters, or a search for the pattern, defined on the whole image plane, which characterizes the axis of rotation is performed. By selecting values which are due only to rotation, the complete rotation is computed, and in the last phase of the procedure every normal flow vector is tested for consistency with the computed motion parameters.

Acknowledgements

I very much thank Prof. Aloimonos for his advice throughout the last years, for his encouragement, criticism and patience. My thanks also go to Barbara Burnett for her expert help in editorial and graphics matters.

References

- [1] G. Adiv, "Determining 3D motion and structure from optical flow generated by several moving objects", *IEEE Trans. Pattern Analysis Machine Intelligence* 7, 384-401, 1985.
- [2] G. Adiv, "Inherent ambiguities in recovering 3D motion and structure from a noisy flow field", in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 70-77, 1985.
- [3] J. Aloimonos and C. Brown, "Direct processing of curvilinear sensor motion from a sequence of perspective images", in *Proc. Workshop on Computer Vision: Representation and Control*, 72-77, 1984.
- [4] S. Barnard and W. Thompson, "Disparity analysis of images", *IEEE Trans. Pattern Analysis Machine Intelligence*, 2, 333-340, 1980.
- [5] A. Bruss and B. Horn, "Passive navigation", *Computer Vision, Graphics, Image Processing* 21, 3-20, 1983.
- [6] P. Burt, J. Bergen, R. Kolczynski, W. Lee, A. Leung, J. Lubin, and H. Shvaytser, "Object tracking with a moving camera", in *Proc. DARPA Image Understanding Workshop*, 2-12, 1989.
- [7] O. Faugeras, *Three Dimensional Computer Vision*. MIT Press, Cambridge, MA, 1992.
- [8] E. Hildreth, *The Measurement of Visual Motion*. MIT Press, Cambridge, MA, 1983.
- [9] B. Horn, *Relative Orientation*. MIT AI Memo 94, 1988.
- [10] B. K. P. Horn and J. Weldon, "Computationally efficient methods for recovering translational motion", in *Proc. Int'l. Conf. on Computer Vision*, 2-11, 1987.
- [11] K. Horn and B. Schunck, "Determining Optical flow", *Artificial Intelligence* 17, 185-203, 1981.
- [12] D. Koller, K. Daniilidis, T. Thorhallson, and H. Nagel, "Model-based object tracking in traffic scenes", in *Proc. Second European Conf. on Computer Vision*, 437-452, 1992.
- [13] H. C. Longuet-Higgins and K. Prazdny, "The interpretation of a moving retinal image", *Proc. Royal Soc. London B* 208, 385-397, 1980.
- [14] H. C. Longuet-Higgins, "A computer algorithm for reconstruction of a scene from two projections", *Nature* 293, 133-135, 1981.
- [15] D. Marr, *Vision*. W.H. Freeman, San Francisco, 1982.
- [16] S. Negahdaripour, *Direct Passive Navigation*. PhD thesis, Department of Mechanical Engineering, MIT, 1986.
- [17] M. E. Spetsakis and J. Aloimonos, "Optimal computing of structure from motion using point correspondence", in *Proc. Int'l. Conf. on Computer Vision*, 449-453, 1988.
- [18] R. Y. Tsai and T. S. Huang, "Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces", *IEEE Trans. Pattern Analysis Machine Intelligence* 6, 13-27, 1984.
- [19] S. Ullman, "The interpretation of structure from motion", *Proc. Royal Soc. London B* 203, 405-426, 1979.
- [20] G. White and E. Weldon, "Utilizing gradient vector distributions to recover motion parameters", in *Proc. IEEE Workshop on Computer Vision*, 132-137, 1987.

3-D Recovery of Structural and Kinematic Parameters from Long Sequences of Noisy Images

Ting-Hu Wu

Signal and Image Processing Institute
Dept. of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089

Rama Chellappa

Dept. of Electrical Engineering
Center for Automation Research and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742

Abstract

This paper presents a feature based approach to estimating the kinematics of a moving camera (or cameras) and the structure of the objects in a stationary environment using long, noisy, monocular and binocular image sequences. Both batch and recursive algorithms are discussed. The image plane coordinates of the feature points in each frame are first detected and then matched over the frames. These noisy image coordinates serve as inputs to our algorithms. Due to the nonlinear nature of perspective projection, a nonlinear least squares method is formulated for the batch algorithm, and a conjugate gradient method is then applied to find the solution. A recursive method using an Iterated Extended Kalman Filter (IEKF) for incremental estimation of motion and structure is also presented. Since the plant model is linear in our formulation, closed form solutions for the state and covariance transition equations are directly derived. Experimental results for several real image sequences are included.

1 Introduction

Visual motion analysis from image sequences is one of the most challenging areas in the field of computer vision. The goal here is to exploit useful 2-D information from images to infer 3-D information about camera motion and scene structure. Apart from its relevance to the understanding of the human visual system, motion analysis has many applications in the areas of target tracking, passive navigation, mobile robotics, missile and autonomous vehicle guidance, and space and underwater exploration. The task at hand is to design and analyze algorithms for recovering 3-D information from 2-D image frames. Many approaches have been advanced by various researchers in the last two decades to address this problem. While the difficulties involved are now better understood, experimental results have had mixed success. By its very nature this problem is ill-posed and, as will be discussed later, the bulk of the approaches investigated have proven to be very sensitive to even moderate levels of error in the image information and in the

calibration of the camera system.

Optical flow and feature-based methods are two major paradigms for motion analysis. Methods that are based on the computation of spatial and temporal image gradients can be broadly classified as optical flow methods [1]–[4]. The optical flow is by definition the apparent movement of the brightness patterns in the image. It is assumed in the previously cited literature to be identical to the motion field (the true 2-D velocity field of each image point), but this is seldom the case. Furthermore the *aperture problem* states that only the component of the optical flow in the direction of the brightness gradient—commonly referred to as the normal flow—can be determined. In order to compute the full optical flow, further arbitrary constraints must be introduced. Finally, the estimation of brightness derivatives is unstable and sensitive to image noise. These limitations have resulted in alternative uses of image brightness derivatives to obtain qualitative environmental information [5], or in direct estimation methods [6], or in methods making exclusive use of normal flow information [7]. The feature-based approach uses discrete image features such as points, lines, or contours as input to the estimation process. The most critical problem for this method is the so called *correspondence problem*: image features must be extracted and matched over the frames both temporally and spatially (in the case of stereoscopic vision). Another problem is the nonlinearity of the equations involved in the algorithm due to the perspective projection model. In this paper, we take a feature-based approach.

Much of the early work on feature-based motion analysis is focused on two- or three-frame problems [8]–[13]. The goal is to determine the transformation between selected feature points observed at two (or three) successive time instants. The resulting equations are usually nonlinear; they are then made linear by increasing the dimensionality of the parameter space. The advantage of this approach is that it does not assume an arbitrary kinematic motion model for the objects being imaged and that conditions for the uniqueness of the parameters to be estimated can be found. The drawback of this method is its sensitivity to the presence of noise [14]–[16]. Also, because the rotation center is implicitly forced to be at the origin of the world coordinate system, the motion parameters computed from every image pair will be different, which makes it impossible to predict the pose

of the camera or of objects at later time instants. It has been shown that increasing the number of feature points only moderately improves the motion-structure estimates.

On the other hand, long-frame based methods can fully utilize the image information and are more robust to noise. Approaches in this category usually involve making an assumption about the nature of the motion model. This requires a certain degree of smoothness in the motion of the imaged objects. The difficulties involved in this method are related to the correspondence problem mentioned earlier and the nonlinearity of the dynamic and observation equations. Thus, methods such as least squares estimation or nonlinear filtering are often adopted to solve these problems.

2 Literature Review

The following is a brief partial review of relevant literature; due to space limitations, many other significant pieces of related work are not mentioned here.

In [17, 18], Gennery used a Kalman-like recursive filter for the tracking of a known 3-D object using line features. Weng et al. [19] introduced a long-frame object motion algorithm based on a Locally Constant Angular Momentum (LCAM) model. Matthies and Shafer [20] proposed a Kalman filter based method of navigating a robot using stereo image sequences. Important work on outdoor vehicle navigation was done by Dickmanns and Graefe [21, 22].

Shariat and Price studied the use of more than two frames for motion analysis assuming that the motion is approximately constant [23]. In [24], Tomasi and Kanade used the singular value decomposition (SVD) technique to factor a matrix of image measurements into two matrices that represent shape and motion, respectively. Zhang and Faugeras [25] developed a method of simultaneously performing motion estimation and object segmentation from a long stereo image sequence using 3-D line segments as features (tokens) and using an Extended Kalman filter (EKF) for motion tracking.

In [26], Broida and Chellappa proposed a Kalman filter-based recursive algorithm for estimating, from a long, noisy, monocular image sequence, the motion and structure of a 2-D object undergoing 2-D motion. This work was extended to 3-D objects and more general kinematic models in [27], where a batch estimation scheme was also presented. Broida et al. also used the Iterated Extended Kalman Filter (IEKF) to effectively implement a recursive algorithm for 3-D motion and structure [28]. A batch algorithm was used to initialize the recursive procedure.

Chandrashekhar and Chellappa [29] developed a moving camera algorithm for passive ranging using a monocular image sequence. Feature points were automatically extracted using Gabor wavelets, and the feature matching process was interleaved with the recursive estimation algorithm, which uses EKF, thereby reducing the search time for finding matching points. In [30], Young and Chellappa proposed a more general motion model which included constant precession and acceleration, and in-

corporated it into their algorithm for stereo image sequences.

3 Outline

For the motion-structure estimation problem, given the perspective projection image model that we use, a nonlinear least squares method is used in the batch algorithms and an IEKF is used in the recursive algorithms. The case of a moving camera and stationary scene, using either monocular or binocular image sequences, is considered. Our approach is to model the motion of the camera(s) as a constant translational and rotational motion using nine motion parameters, namely the 3-D vectors of the position of the rotation center and its linear and angular velocities. The structure parameters are the 3-D coordinates of the salient feature points in the inertial coordinate system.

The justification for choosing this motion model is the smoothness of the object motion. As long as the sampling rate is high enough, the object motion can be approximated over a short period of time using only a first order motion model; deviations from this model can be treated as noise which can be taken care of later by the recursive (tracking) filter. A standard rotation matrix is used to describe the rotational motion rather than the quaternion representation used earlier. Under these conditions, linear plant models can be obtained and closed form solutions for the state and covariance transition differential equations can be directly derived without the need for a time-consuming numerical integration step.

To handle the correspondence problem, the Gabor wavelet method is used to extract salient feature points from each image [31], and matching and tracking of these points are performed using the method originally proposed in [32]. These image correspondences are then used as inputs to our algorithms. The noise in the data includes quantization error, detection error, system (camera) parameter error, etc. Due to the motion of the camera(s), some feature points may be outside the image or temporarily occluded by other objects in the scene in some image frames; this is known as the occlusion problem. In our batch algorithms this problem is handled by omitting the measurements of the missing points in the least squares criterion functions. Similarly, in the recursive algorithms we incorporate only the measurements obtained from the unoccluded points into the measurement equations.

For binocular imagery, the traditional stereo triangulation method fails when the images from the two cameras are not taken at the same time. For our binocular algorithm, however, since asynchronism is allowed, the two cameras can function independently.

4 Monocular Imagery

The next two sections briefly explore the motion-structure estimation problem for the case of a dynamic observer moving in a static environment. For detailed derivations and more experimental results, the interested reader is referred to [33, 34].

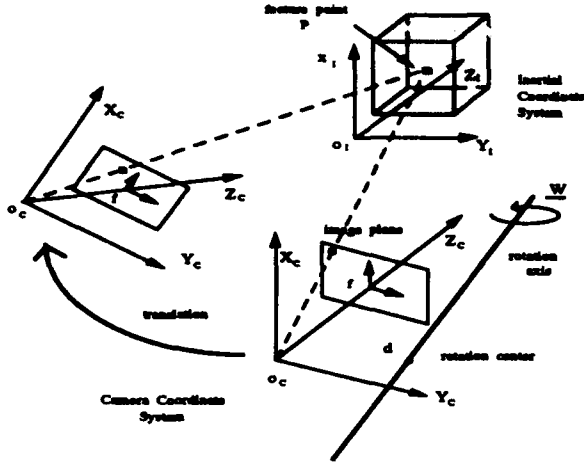


Figure 1: Monocular imaging and motion models of the moving camera.

4.1 Image Model

As shown in Figure 1, a camera is moving with respect to a fixed environment. A 3-D inertial (world) coordinate system I is fixed on the ground, and the camera coordinate system C is fixed on the camera with its z -axis pointing along the optical axis. The focal length is f , so the image plane is $z = f$ in the C coordinate system. Suppose that at time t the i^{th} feature point P_i has 3-D coordinates

$$\underline{s}_{Ci}(t) \equiv (s_{Cix}(t), s_{Ciy}(t), s_{Ciz}(t))^T$$

in coordinate system C . By the central projection model, the coordinates on the image plane can be represented as

$$\begin{cases} X_i(t) = f \frac{s_{Cix}(t)}{s_{Ciz}(t)} + n_{Xi}(t) \\ Y_i(t) = f \frac{s_{Ciy}(t)}{s_{Ciz}(t)} + n_{Yi}(t) \end{cases} \quad (1)$$

where the n 's are the additive noise variables.

The coordinate systems C and I are set to be coincident at time $t = t_0$. As time proceeds, the system C starts to move with the camera and the system I is left behind, fixed on the ground. The rotation center is assumed to be fixed on the rotation axis in system C at all times. Let \underline{d} be the coordinates of the rotation center in C , and denote the trajectory of this center in I at time t by

$$\underline{r}_I(t) \equiv (r_{Ix}(t), r_{Iy}(t), r_{Iz}(t))^T$$

Since coordinate system C coincides with coordinate system I at $t = t_0$, we also have

$$\underline{r}_I(t_0) = \underline{d}$$

4.2 Motion Model

Under the assumption of constant translational motion, the camera velocity \underline{v} can be expressed as

$$\underline{v} \equiv \frac{d}{dt} \underline{r}_I(t) = \dot{\underline{r}}_I \quad (2)$$

Integrating Equation (2), we have

$$\underline{r}_I(t) = \underline{d} + (t - t_0) \underline{v} \quad (3)$$

The six time-invariant components of \underline{d} and $\dot{\underline{r}}_I$ are called the *translational motion parameters*.

Let $\underline{\omega}$ be the constant angular velocity of the camera relative to the inertial coordinate system I . The vector representations of this quantity in C and I are the same because we have already assumed that C and I coincide at time $t = t_0$. These three time-invariant components of $\underline{\omega}$ are the *rotational motion parameters* that we want to estimate.

Consider another coordinate system C' whose origin is at \underline{d} and which has the same directional vectors along the x -, y - and z -axes as system C at all times. The camera is rotating about the rotation center (whose location is \underline{d}) with constant angular velocity $\underline{\omega}$. If the i^{th} feature point P_i has coordinates \underline{s}_{Ii} in I , decomposition of rotation and translation gives

$$\begin{aligned} \underline{s}_{Ii} &\equiv (s_{Iix}, s_{Iiy}, s_{Iiz})^T \\ &= \underline{r}_I(t) + R(\underline{\omega}, t - t_0) \cdot \underline{s}_{C'i}(t) \end{aligned} \quad (4)$$

Let us denote system C' at time t by $C'(t)$; then at time t , vector $\underline{s}_{C'(t_0)i}$ rotates to $\underline{s}_{C'(t)i}$. Since vector $\underline{s}_{C'(t_0)i}$ in $C'(t_0)$ has the same coordinates as vector $\underline{s}_{C'(t)i}$ in $C'(t)$, and the total angle rotated is $|\underline{\omega}|(t - t_0)$, the coordinates of $\underline{s}_{C'(t)i}$ in $C'(t_0)$ can be shown to be

$$\begin{aligned} &\underline{s}_{C'(t)i} \text{ in } C'(t_0) \\ &= R_0 \left[\frac{w_x}{|\underline{\omega}|}, \frac{w_y}{|\underline{\omega}|}, \frac{w_z}{|\underline{\omega}|}; |\underline{\omega}|(t - t_0) \right] \underline{s}_{C'i}(t) \end{aligned} \quad (5)$$

where $R_0[n_1, n_2, n_3; \theta]$ is the standard rotation matrix which rotates a 3-D vector representation by the angle θ with respect to the unit directional vector $(n_1, n_2, n_3)^T$. Also, $C'(t)$ is a shifted version of $C(t)$, so

$$\underline{s}_{C'i}(t) = \underline{s}_{Ci}(t) - \underline{d}$$

Using Equations (3) and (5), Equation (4) can be written as

$$\begin{aligned} \underline{s}_{Ii} &\equiv (s_{Iix}, s_{Iiy}, s_{Iiz})^T \\ &= \underline{r}_I(t) + R(\underline{\omega}, t - t_0) \cdot [\underline{s}_{Ci}(t) - \underline{d}] \\ &= \underline{d} + (t - t_0) \underline{v} + R(\underline{\omega}, t - t_0) \cdot [\underline{s}_{Ci}(t) - \underline{d}] \end{aligned} \quad (6)$$

where

$$R(\underline{\omega}, t - t_0) = R_0 \left[\frac{w_x}{|\underline{\omega}|}, \frac{w_y}{|\underline{\omega}|}, \frac{w_z}{|\underline{\omega}|}; |\underline{\omega}|(t - t_0) \right] \quad (7)$$

Rearranging Equation (6), we get

$$\begin{aligned} &(s_{Cix}(t), s_{Ciy}(t), s_{Ciz}(t))^T \\ &= \underline{d} + R^{-1}(\underline{\omega}, t - t_0) \cdot [\underline{s}_{Ii} - \underline{d} - (t - t_0) \dot{\underline{r}}_I] \end{aligned} \quad (8)$$

where $R^{-1}(\underline{\omega}, t - t_0) = R^T(\underline{\omega}, t - t_0) = R(\underline{\omega}, -(t - t_0))$.

In our algorithms, the three time-invariant components of each \underline{s}_{Ii} are called the *structure parameters*.

4.3 Batch Formulation

In this formulation, the unrecoverable global scaling factor is taken care of by normalizing the translational and structure parameters by, say, the z -component of the M^{th} feature point, s_{IMz} . Then these normalized parameters can be expressed as

$$\begin{aligned} \underline{d}^N &\equiv (d_x^N, d_y^N, d_z^N)^T \\ &\equiv \left(\frac{d_x}{s_{IMz}}, \frac{d_y}{s_{IMz}}, \frac{d_z}{s_{IMz}} \right)^T \\ \underline{v}^N &\equiv (v_x^N, v_y^N, v_z^N)^T \\ &\equiv \left(\frac{v_x}{s_{IMz}}, \frac{v_y}{s_{IMz}}, \frac{v_z}{s_{IMz}} \right)^T \\ \underline{s}_{Ii}^N &\equiv (s_{Iix}^N, s_{Iiy}^N, s_{Iiz}^N)^T \\ &\equiv \left(\frac{s_{Iix}}{s_{IMz}}, \frac{s_{Iiy}}{s_{IMz}}, \frac{s_{Iiz}}{s_{IMz}} \right)^T \end{aligned} \quad (9)$$

In order to remove the redundant degree of freedom caused by random shift of the rotation center, we set, say, $d_z = 0$ in our algorithms. Using Equation (9), Equation (8) can be written as

$$\begin{aligned} &(s_{Cix}(t), s_{Ciy}(t), s_{Ciz}(t))^T \\ &= s_{IMz} \left[\underline{d}^N + R^{-1}(\underline{w}, t-t_0) (\underline{s}_{Ii}^N - \underline{d}^N - (t-t_0) \underline{v}^N) \right] \end{aligned} \quad (10)$$

Equation (1) thus becomes

$$X_i(t) = f \frac{d_x^N + R_1^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ii}^N - \underline{d}^N - (t-t_0) \underline{v}^N]}{R_3^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ii}^N - \underline{d}^N - (t-t_0) \underline{v}^N]} + n_{X_i}(t) \quad (11)$$

$$Y_i(t) = f \frac{d_y^N + R_2^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ii}^N - \underline{d}^N - (t-t_0) \underline{v}^N]}{R_3^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ii}^N - \underline{d}^N - (t-t_0) \underline{v}^N]} + n_{Y_i}(t)$$

where R_i^{-1} is the i^{th} row vector of R^{-1} and d_x^N is set to zero in vector \underline{d}^N .

The $3M + 7$ unknown motion and structure parameters for motion estimation from a sequence of monocular images are chosen as

$$\underline{y}^N \equiv (d_x^N, d_y^N, \underline{v}^N, \underline{w}, \underline{s}_{I1}^N, \dots, \underline{s}_{I(M-1)}^N, s_{IMz}^N, s_{IMy}^N)^T.$$

The least squares estimate of the motion and structure parameters \underline{y}^N for Equation (11) is obtained by finding the minimum of the following cost function:

$$\begin{aligned} &\min_{\underline{y}^N} \sum_i \sum_{k=1}^n \\ &\left\{ \left[X_{j_k}(t_i) - f \frac{d_x^N + R_1^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ij_k}^N - \underline{d}^N - (t-t_0) \underline{v}^N]}{R_3^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ij_k}^N - \underline{d}^N - (t-t_0) \underline{v}^N]} \right]^2 \right. \\ &\left. + \left[Y_{j_k}(t_i) - f \frac{d_y^N + R_2^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ij_k}^N - \underline{d}^N - (t-t_0) \underline{v}^N]}{R_3^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ij_k}^N - \underline{d}^N - (t-t_0) \underline{v}^N]} \right]^2 \right\} \end{aligned} \quad (12)$$

If the answer obtained by our algorithm is $\underline{d} = (d_x, d_y, 0)$, then any vector $\tilde{\underline{d}}$ that has the form $\tilde{\underline{d}} = \underline{d} + \alpha \underline{w}$ (where α is some constant) will also satisfy Equation (8). This is because \underline{w} is an eigenvector of rotation

matrix R (and also of R^{-1}) with eigenvalue 1. This also confirms the fact that any point along the rotation axis can serve as the rotation center without affecting the image plane coordinates.

4.4 Recursive Formulation

In Section 4.3, we set $d_z = 0$; thus $r_{Iz}^N(t) = (t-t_0) r_{Iz}^N$, so that $r_{Iz}^N(t)$ can be dropped out of the estimation process. The state vector $\underline{x}^N(t)$ chosen for the recursive algorithm thus consists of the following normalized motion and structure parameters:

$$\underline{x}^N(t) \equiv (r_{Ix}^N(t), r_{Iy}^N(t), \dot{\underline{r}}_I^N, \underline{w}, \underline{s}_{I1}^N, \dots, s_{IMz}^N, s_{IMy}^N)^T.$$

Plant Equation

Under the assumption of constant translational and angular velocity, the time derivatives of $\dot{\underline{r}}_I^N$, \underline{w} , \underline{s}_{I1}^N , $\underline{s}_{I2}^N, \dots, \underline{s}_{IM}^N$ are all zero. Using this fact and Equation (2), we get the linear plant equation

$$\dot{\underline{x}}^N(t) = F^N \underline{x}^N(t) \quad (13)$$

where F^N is the sparse square matrix

$$F^N \equiv \{ F_{13}^N = F_{24}^N = 1; \text{ all other elements } F_{ij}^N = 0 \}. \quad (14)$$

Measurement Equation

Let

$$\begin{aligned} \underline{z}^N(t) &\equiv \left(\frac{r_{Ix}(t)}{s_{IMz}}, \frac{r_{Iy}(t)}{s_{IMz}}, \frac{(t-t_0) \dot{r}_{Iz}}{s_{IMz}} \right)^T \\ &\equiv (r_{Ix}^N(t), r_{Iy}^N(t), (t-t_0) \dot{r}_{Iz}^N)^T \end{aligned}$$

Substituting this into Equation (11), we obtain the measurement equation

$$\underline{z}^N(t_i) = \underline{h}^N[\underline{x}^N(t_i); t_i] + \underline{v}^N(t_i) \quad (15)$$

where

$$\underline{z}^N(t_i) = \begin{pmatrix} X_{j_1}(t_i) \\ Y_{j_1}(t_i) \\ \vdots \\ X_{j_n}(t_i) \\ Y_{j_n}(t_i) \end{pmatrix}, \quad \underline{v}^N(t_i) = \begin{pmatrix} n_{X_{j_1}}(t_i) \\ n_{Y_{j_1}}(t_i) \\ \vdots \\ n_{X_{j_n}}(t_i) \\ n_{Y_{j_n}}(t_i) \end{pmatrix}$$

and

$$\begin{aligned} &\underline{h}^N[\underline{x}^N(t_i); t_i] \\ &= \begin{pmatrix} f \frac{r_{Ix}^N(t_i) - (t_i-t_0) \dot{r}_{Ix}^N + R_1^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ij_1}^N - \underline{r}^N(t_i)]}{R_3^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ij_1}^N - \underline{r}^N(t_i)]} \\ f \frac{r_{Iy}^N(t_i) - (t_i-t_0) \dot{r}_{Iy}^N + R_2^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ij_1}^N - \underline{r}^N(t_i)]}{R_3^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ij_1}^N - \underline{r}^N(t_i)]} \\ \vdots \\ f \frac{r_{Ix}^N(t_i) - (t_i-t_0) \dot{r}_{Ix}^N + R_1^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ij_n}^N - \underline{r}^N(t_i)]}{R_3^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ij_n}^N - \underline{r}^N(t_i)]} \\ f \frac{r_{Iy}^N(t_i) - (t_i-t_0) \dot{r}_{Iy}^N + R_2^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ij_n}^N - \underline{r}^N(t_i)]}{R_3^{-1}(\underline{w}, t-t_0) [\underline{s}_{Ij_n}^N - \underline{r}^N(t_i)]} \end{pmatrix} \end{aligned}$$

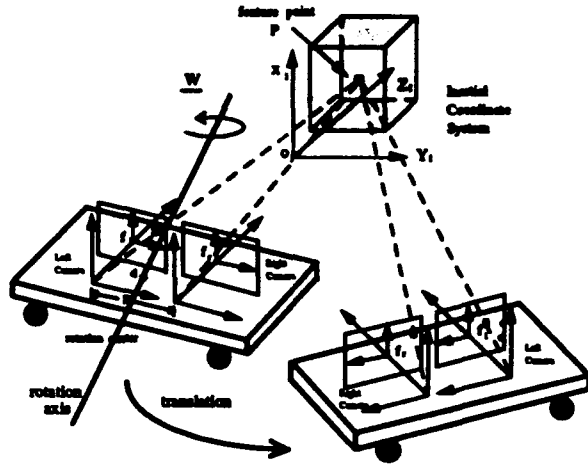


Figure 2: Binocular imaging and motion models of the moving cameras.

where j_1, j_2, \dots, j_n are unoccluded feature points at time t_i . Therefore, only unoccluded points are incorporated in the measurement equation (15).

Since our plant model is linear, and matrix F^N is sparse, closed form solutions for the state and covariance transition equations can be derived directly without a numerical integration step.

State Transition Equation

$\dot{x}^N, \underline{u}, \underline{x}_{j1}^N, \underline{x}_{j2}^N, \dots, \underline{x}_{jM}^N$ are all constant in t under the rigidity assumption and the model of constant translational and angular velocities; this together with (3) gives the state transition equation

$$\underline{x}^N(t_{i+1}^-) = [I + (t_{i+1}^- - t_i^+) F^N] \underline{x}^N(t_i^+) \quad (16)$$

where I is the identity matrix.

Covariance Transition Equation

By (14), we see that $(F^N)^2$ is a zero matrix. Direct substitution may then be used to verify that the covariance transition equation has the form

$$P^N(t_{i+1}^-) = [I + (t_{i+1}^- - t_i^+) F^N] P^N(t_i^+) [I + (t_{i+1}^- - t_i^+) F^N]^T \quad (17)$$

5 Binocular Imagery

5.1 Image and Motion Models

As shown in Figure 2, the two cameras are installed on a platform which is moving in a stationary environment. Two camera coordinate systems with the same orientation, LC and RC , are attached to the left and right cameras, respectively, with their z -axes coinciding with the optical axes of the cameras. A 3-D inertial coordinate system I is chosen to be coincident with the left camera coordinate system at time $t = t_0$, without loss of generality. As time proceeds, the platform moves and the inertial coordinate system is left behind, fixed on the ground.

Let a point have 3-D coordinates \underline{x}_I in inertial coordinate system I , and let its coordinate representations

in the left and right camera coordinate systems at time t be $\underline{x}_{RC}(t)$ and $\underline{x}_{LC}(t)$, respectively. Suppose that the two cameras are linked together; then the relative orientation and the displacement vector between the cameras remain unchanged at all times. Thus, the transformation between these systems can be expressed as

$$\begin{aligned} \underline{x}_{RC}(t) &= \underline{x}_{LC}(t) - D \\ \underline{x}_{LC}(t_0) &= \underline{x}_I \end{aligned} \quad (18)$$

where D is the displacement vector from the origin of the LC coordinate system to that of the RC system and is assumed to be known.

Suppose that at time t the i th feature point of the scene, P_i , has 3-D coordinates

$$\underline{x}_{LCi}(t) \equiv (s_{LCis}(t), s_{LCiy}(t), s_{LCiz}(t))^T$$

and

$$\underline{x}_{RCi}(t) \equiv (s_{RCis}(t), s_{RCiy}(t), s_{RCiz}(t))^T$$

in the LC and RC systems, respectively. Then, by the central projection model, the coordinates of the images of P_i on the left and right image planes can be represented as

$$\begin{cases} X_{li}(t) = f_l \frac{s_{LCis}(t)}{s_{LCiz}(t)} + n_{Xli}(t) \\ Y_{li}(t) = f_l \frac{s_{LCiy}(t)}{s_{LCiz}(t)} + n_{Yli}(t) \end{cases} \quad (19)$$

and

$$\begin{cases} X_{ri}(t) = f_r \frac{s_{RCis}(t)}{s_{RCiz}(t)} + n_{Xri}(t) \\ Y_{ri}(t) = f_r \frac{s_{RCiy}(t)}{s_{RCiz}(t)} + n_{Yri}(t) \end{cases} \quad (20)$$

where f_l and f_r are the focal lengths of the two cameras and the n 's are additive noise processes.

Similarly, the motion of the platform can be decomposed into a rotation about the rotation center and a translation of this center. The rotation center is chosen to be fixed on the platform and has coordinates \underline{d} in the LC coordinate system at all times. Denote the trajectory of the rotation center at time t in I by

$$\underline{r}_I(t) \equiv (r_{Ix}(t), r_{Iy}(t), r_{Iz}(t))^T$$

Since systems I and LC coincide at time $t = t_0$, we have

$$\underline{r}_I(t) = \underline{d} + (t - t_0) \cdot \underline{v} \quad (21)$$

where $\underline{v} \equiv \dot{\underline{r}}_I$ is the translational velocity.

Suppose the platform undergoes constant translational motion \underline{v} as well as constant angular motion $\underline{\omega}$ with respect to system I . Following the same procedure as in the monocular case, the transformations between the LC and RC systems and system I are found to be

$$\begin{aligned} (s_{LCis}(t), s_{LCiy}(t), s_{LCiz}(t))^T \\ = \underline{d} + R^{-1}(\underline{\omega}, t - t_0) \cdot [\underline{x}_{Ii} - \underline{d} - (t - t_0) \underline{v}] \end{aligned} \quad (22)$$

and

$$\begin{aligned} (s_{RCis}(t), s_{RCiy}(t), s_{RCiz}(t))^T \\ = \underline{d} - \underline{D} + R^{-1}(\underline{\omega}, t - t_0) \cdot [\underline{x}_{Ii} - \underline{d} - (t - t_0) \underline{v}] \end{aligned} \quad (23)$$

The three components of $\underline{\omega}$ are the *rotational motion parameters* and the six components of \underline{d} and \underline{v} are the *translational motion parameters*, while the three components of each \underline{x}_{Ii} are the *structure parameters*. Again, these components are all time-invariant under our model setup.

5.2 Batch and Recursive Formulations

The $3M + 8$ unknown motion and structure parameters for motion estimation from a sequence of stereo images are chosen as

$$\underline{y} \equiv (d_x, d_y, \underline{v}, \underline{w}, \underline{z}_{I1}, \dots, \underline{z}_{IM})^T$$

Using Equations (22) and (23), Equations (19) and (20) become

$$X_{li}(t) = f_i \frac{d_x + R_i^{-1}(\underline{w}, t-t_0) \cdot [\underline{z}_{li} - \underline{d} - (t-t_0)\underline{v}]}{R_i^{-1}(\underline{w}, t-t_0) \cdot [\underline{z}_{li} - \underline{d} - (t-t_0)\underline{v}]} + n_{Xli}(t) \quad (24)$$

$$Y_{li}(t) = f_i \frac{d_y + R_i^{-1}(\underline{w}, t-t_0) \cdot [\underline{z}_{li} - \underline{d} - (t-t_0)\underline{v}]}{R_i^{-1}(\underline{w}, t-t_0) \cdot [\underline{z}_{li} - \underline{d} - (t-t_0)\underline{v}]} + n_{Yli}(t)$$

and

$$X_{ri}(t) = f_r \frac{d_x - D_x + R_i^{-1}(\underline{w}, t-t_0) \cdot [\underline{z}_{li} - \underline{d} - (t-t_0)\underline{v}]}{-D_x + R_i^{-1}(\underline{w}, t-t_0) \cdot [\underline{z}_{li} - \underline{d} - (t-t_0)\underline{v}]} + n_{Xri}(t)$$

$$Y_{ri}(t) = f_r \frac{d_y - D_y + R_i^{-1}(\underline{w}, t-t_0) \cdot [\underline{z}_{li} - \underline{d} - (t-t_0)\underline{v}]}{-D_y + R_i^{-1}(\underline{w}, t-t_0) \cdot [\underline{z}_{li} - \underline{d} - (t-t_0)\underline{v}]} + n_{Yri}(t)$$

where R_i^{-1} is the i^{th} row vector of R^{-1} and d_x is set to zero in vector \underline{d} .

The least squares estimate of the motion and structure parameters \underline{y} of Equation (24) is obtained by finding the minimum of the cost function, which has a similar form to that in Equation (12).

In the recursive formulation, $\underline{r}_I(t)$ is chosen as one of the states instead of \underline{d} , to account for the dynamic property of the Kalman filter. Since d_x is set to zero to fix the rotation center, by (21) we have $r_{Ix}(t) = (t-t_0)v_x$. Thus if v_x is included in the state vector, $r_{Ix}(t)$ is no longer needed in the estimation process. Hence, the state vector chosen for the recursive algorithm consists of the following motion and structure parameters:

$$\underline{x}(t) \equiv (r_{Ix}(t), r_{Iy}(t), \underline{r}_I, \underline{w}, \underline{z}_{I1}, \dots, \underline{z}_{IM})^T.$$

The plant, measurement, state and covariance transition equations can be derived similarly to their derivation for the monocular case. After these equations are obtained, a standard IEKF can be used.

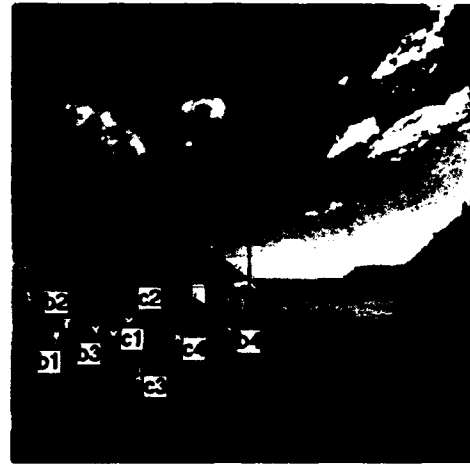
Detailed proofs of the uniqueness of the estimated parameters in the binocular case can be found in [33].

6 Experimental Results

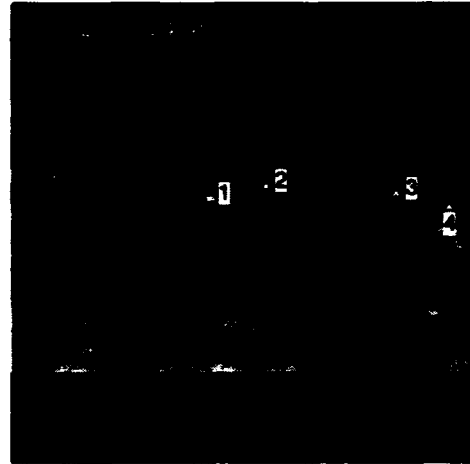
This section describes real imagery experiments for both the monocular and binocular cases. For results on simulated imagery and additional real imagery experiments, see [33].

6.1 Monocular Imagery

Two real image sequences were tested. The inputs to our algorithms are the 2-D image coordinates of the salient feature points detected using the algorithm proposed in [32]. In this algorithm, the global image change due to unknown camera motion is first compensated by an image registration algorithm [35], which automatically detects feature points and estimates the rotation, translation and scaling between two images. The feature points are then tracked over the image sequence using a weighted cross-correlation match method. The method



(a)



(b)

Figure 3: Locations of the feature points in the first frame of each sequence: (a) the UMASS Rocket sequence; (b) the NASA helicopter (ARC) sequence.

can only find the best matches at grid points; to obtain subpixel accuracy feature point correspondences, a feature point in the current frame is first transformed to the coordinates of the next frame after the motion of the camera has been computed using the image registration algorithm. The four grid points nearest to this feature point are found and their best grid point matches in the next frame are located. An interpolation scheme is then applied to these four grid points to get the subpixel accuracy correspondences. These correspondences serve as the input to our algorithms. The detailed implementation of this point correspondence algorithm is described in [32].

In our experiments, for comparison purposes, feature points with 3-D ground truth are hand-picked in the first frame of the UMASS Rocket sequence. The point correspondence algorithm [32] is then used for the subsequent frames. For the NASA sequence, feature points were automatically detected and tracked in all the frames. The locations of the feature points used for each sequence are marked in Figure 3.

The world coordinate system is set to be the first-frame image coordinate system with the origin located at the center of the image plane. The x -axis points to the right, the y -axis points upward, and by the right hand rule, the z -axis thus points out of the image plane. The initial guesses of the parameters for the batch algorithm are all 0.001.

6.1.1 UMASS Rocket Sequence

The 30-frame UMASS "Rocket" ALV sequence [36] is used in this section; the first and the twenty-first images are shown in Figure 4 (a) and (b). Eight feature points with known ground truth were used. Experiments with the batch algorithm were based on four feature points. The computed motion and structure parameters and the normalized structure ground truth of these points in the first frame image coordinate system are shown in Tables 1 and 2.

Table 1: Structure estimates for the UMASS Rocket sequence.

	Feature point	Normalized structure ground truth			Estimated structure		
Box 1-4	1	0.494	0.317	0.866	0.489	0.327	0.901
	2	0.601	0.383	1.153	0.563	0.367	1.109
	3	0.425	0.357	0.965	0.420	0.349	1.003
	4	0.020	0.342	1.0	-0.001	0.351	1.0
Cone 1-4	1	0.365	0.345	1.015	0.354	0.341	0.995
	2	0.433	0.433	1.394	0.457	0.456	1.456
	3	0.169	0.250	0.592	0.161	0.248	0.572
	4	0.165	0.341	1.0	0.162	0.347	1.0

Table 2: Motion parameter estimates for the UMASS Rocket sequence.

	Estimated rotational velocity			Estimated translational velocity		
Cone	-0.00017	0.01027	0.00021	0.00253	-0.00800	0.03347
Box	0.00414	-0.00341	0.00013	0.01251	-0.00683	0.04216

From the image sequence, we see that there is almost no rotational motion and the camera appears to be moving along a straight line going leftward and into the image plane. Thus the z -component of the translational motion should be the largest one, followed by the x -component, and the y -component should be very close to zero. (Recall that the depths of the feature points are all negative, so that after the normalization step, the z - and x -components of the translational velocity should be positive.) According to these observations, the translational velocity computed in Table 2 for the box seems to be accurate.

In the recursive experiment, the set of four cone points in Table 1 was used. Because one of these points disappears after frame 6, one disappears after frame 10, and all the others are outside the image after frame 21, it was not possible to apply the normally used recursive algorithm. Instead, the output from the six-frame batch algorithm was used as the initial guess, and the Kalman filter runs from frame 1 to frame 21, acting like a non-causal smoother through the first six frames. Since motion ground truth is not available, only the computed

structure error percentages are shown in Figure 5 (due to space limitations, only points 1 and 3 are shown).

Since the camera was not calibrated in this experiment, we also tried to estimate the field of view of the camera as well as the center of the image. It was found that the fields of view of the camera were very close to the given specification, but the center of the image was estimated to be at (240,306) instead of the assumed location (256,270). This is because the original size of each image is 512×484 , and our location of the origin is at the lower left corner of a 512×512 image plane. The two poles in the far field were also tested, but since they are close to the FOE and their depths are large, one- or two-pixel errors introduced by the feature detection algorithm can cause fairly large amounts of error in the structure estimates.

6.1.2 NASA Sequence

A ten-frame NASA Helicopter (ARC) sequence, as shown in Figure 4 (c) and (d), was also used. Since this sequence is too short to apply the recursive algorithm, only the batch algorithm was tested. Also, ground truth information and camera calibration parameters were not available, so in running the batch algorithm, the field of view was assumed to be 40 degrees and the optical center was assumed to coincide with the center of each image. The estimates of the structure and motion parameters are shown in Tables 3 and 4.

Table 3: Structure estimates for the NASA Helicopter sequence

Feature point	Estimated structure		
1	0.036887	-0.045965	0.971495
2	-0.047572	-0.079270	1.071193
3	-0.242800	-0.066243	1.002631
4	-0.326975	-0.045814	1.0

Table 4: Motion parameter estimates for the NASA Helicopter sequence

Estimated rotational velocity			Estimated translational velocity		
-0.0002	0.0194	-0.0058	-0.0385	-0.0043	0.0405

6.2 Binocular Imagery

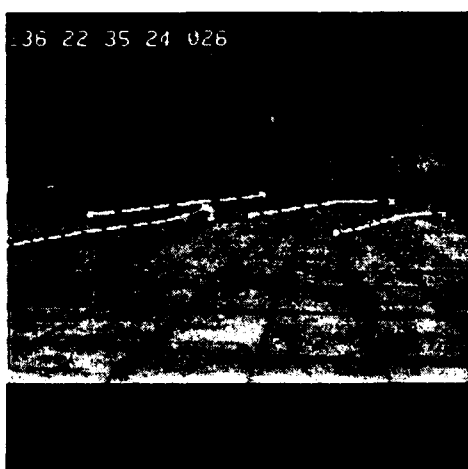
The 10-frame "Forward" stereo image sequence is used in this section. The first and last image pairs are shown in Figure 6 with the feature trajectories superimposed on them. The inertial coordinate system is taken to be the first-frame left camera coordinate system with its origin located at (246.6, 225.6). The x -axis points to the right, the y -axis points downward, and by the right hand rule, the z -axis thus points into the image. The displacement vector \underline{D} is (1.0, 0, 0) inch, while the motion of the platform is 0.2 inches/frame straight ahead (pure translation). Structure information was not available to us.



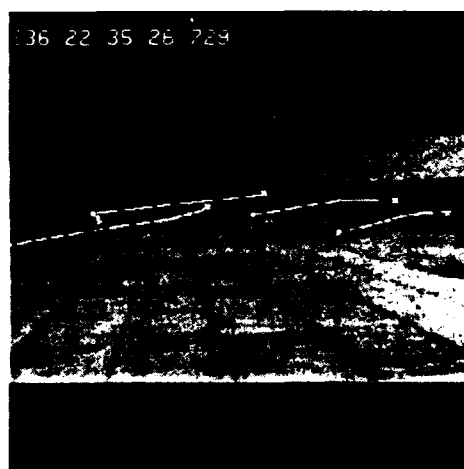
(a)



(b)

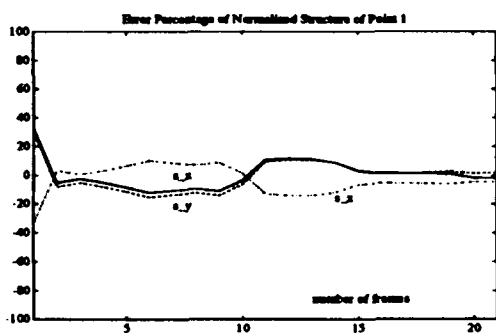


(c)

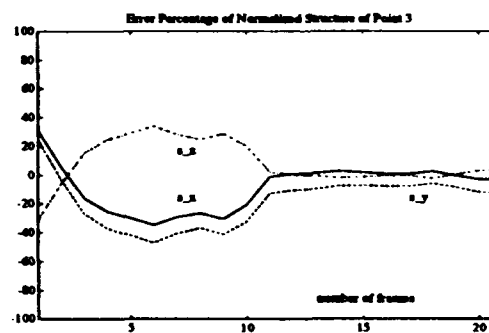


(d)

Figure 4: Image plane trajectories of feature points: (a), (b) first and 21st frames of the UMASS Rocket sequence; (c), (d) first and last frames of the NASA sequence.



(a)



(b)

Figure 5: Normalized structure error percentages for the UMASS Rocket sequence computed by the Kalman filter (cone): (a) point 1; (b) point 3.

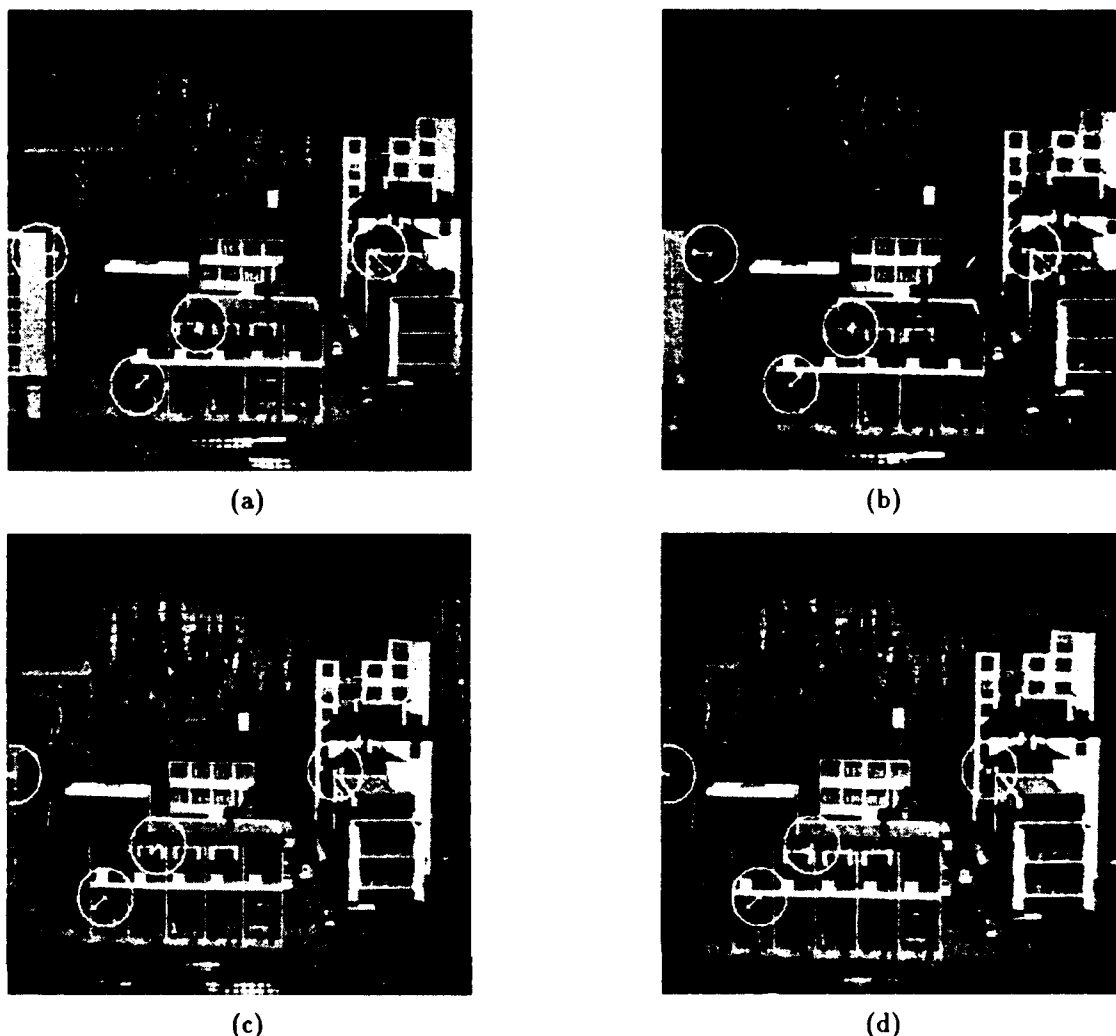


Figure 6: Image plane trajectories of feature points (in circled areas) in the Forward sequence: (a) first frame of left sequence, (b) last frame of left sequence, (c) first frame of right sequence, (d) last frame of right sequence.

Four feature points are used to test both the batch and recursive algorithms; their locations are shown in Figure 7. These feature points are first detected and tracked over the frames of the left image sequence using the method proposed in [32]. Then the first frame of the left sequence is registered with the first frame of the right sequence to find the corresponding feature points in these two frames [35]. The correspondences of the feature points in the first frames of the left and right sequences are shown in Figure 8. After the matching points in the first frame of the right sequence are found, the same tracking algorithm can be used for the right image sequence. The coordinates of these feature points then serve as the input to our algorithms. The estimated motion and structure parameters using the 10-frame batch method are listed in Table 5. The error in the velocity along the z -direction is around 8 percent. The outputs of the 2-frame batch method were used as the inputs to our recursive algorithm; the results for the estimated velocity are shown in Figure 9.

Table 5: Structure and motion parameter estimates for the Forward sequence.

Feature point	Estimated structure	Estimated velocities		
		Rotational (radians)		
1	-4.414 0.527 22.338	-0.0002	-0.0002	-0.0001
2	-2.170 2.953 22.588			
3	-0.804 1.916 22.372			
4	4.168 0.542 26.788	0.0052	0.0009	0.1848

7 Conclusion

Complete, automatic algorithms, starting from feature correspondences and yielding motion and structure estimates, are reported in this paper. Both the batch and recursive algorithms for motion and structure recovery are found to be robust in spite of different sources and levels of noise. Our algorithms perform well even when as few as four feature correspondences are used; this fact, together with the ability to handle occlusion, makes it possible to handle appearances and disappearances of features as well as relatively short-span observations of features, thus reducing the computational cost associ-

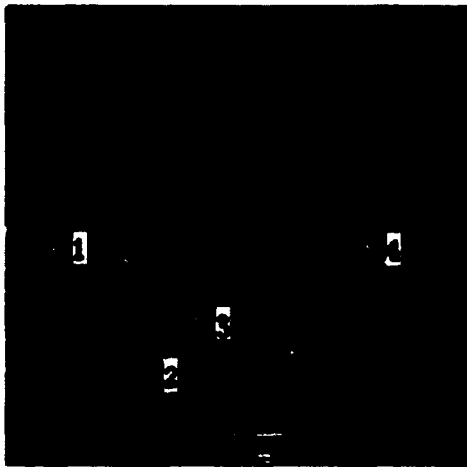


Figure 7: Locations of feature points in the first frame of the left image sequence.

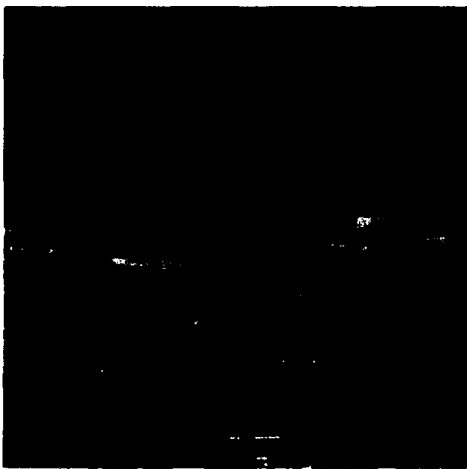


Figure 8: Corresponding feature points in the left and right image sequences superimposed on the first frame of the left sequence.

ated with maintaining feature correspondences. In the binocular case, if images are taken at the same time by both cameras, stereo triangulation can give us good initial guesses for the structure parameters to speed up the convergence of both the batch and recursive algorithms. With little modification, the algorithms can also be applied to situations where the two cameras undergo different motions.

Acknowledgement

The authors are grateful to the computer vision research group at the University of Massachusetts for providing the Rocket Sequence. The NASA Sequence was made available through the courtesy of NASA Ames Research Center. We also wish to thank Dr. Larry Matthies of the Jet Propulsion Laboratory for providing the Forward Stereo Sequence.

References

- [1] G. Adiv, "Determining Three-Dimensional Motion

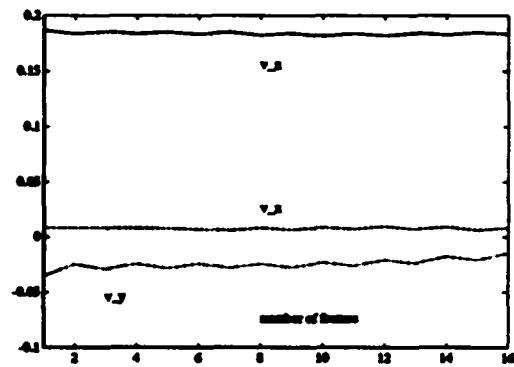


Figure 9: Translational velocity for the Forward sequence computed by the Kalman filter. The even frame numbers are for the right image sequence while the odd numbers are for the left image sequence.

and Structure from Optical Flow Generated by Several Moving Objects," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-7, pp. 384-401, July 1985.

- [2] D.H. Ballard and O.A. Kimball, "Rigid Body Motion from Depth and Optical Flow," *Comput. Vision, Graph., Image Processing*, vol. 22, pp. 95-115, April 1983.
- [3] B.K.P. Horn and B.G. Schunck, "Determining Optical Flow," *Artificial Intelligence*, vol. 17, pp. 185-203, Aug. 1981.
- [4] W.B. Thompson, K.M. Mutch, and V.A. Berzins, "Dynamic Occlusion Analysis in Optical Flow Fields," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-7, pp. 374-383, July 1985.
- [5] T. Poggio, A. Verri, and V. Torre, "Green Theorems and Qualitative Properties of the Optical Flow," AI Memo 1289, MIT, April 1991.
- [6] J. Heel, "Direct Estimation of Structure and Motion from Multiple Frames," AI Memo 1190, MIT, March 1990.
- [7] J.Y. Aloimonos, Z. Duric, C. Fermüller, L. Huang, E. Rivlin, and R. Sharma, "Behavioral Visual Motion Analysis," in *Proc. DARPA Image Understanding Workshop*, San Diego, CA, pp. 521-533, Jan. 1992.
- [8] H.C. Longuet-Higgins, "The Reconstruction of a Scene from Two Projections—Configurations that Defeat the 8-point Algorithm," in *Proc. IEEE Conf. on Artificial Intelligence Applications*, Denver, CO, pp. 395-397, Dec. 1984.
- [9] J. Roach and J. Aggarwal, "Determining the Movement of Objects from a Sequence of Images," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-2, pp. 554-562, Nov. 1980.
- [10] R.Y. Tsai and T.S. Huang, "Estimating Three-Dimensional Motion Parameters of a Rigid Planar Patch," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-29, pp. 1147-1152, Dec. 1981.

- [11] R.Y. Tsai, T.S. Huang, and W.L. Zhu, "Estimating Three-Dimensional Motion Parameters of a Rigid Planar Patch, II: Singular Value Decomposition," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-30, pp. 525-534, Aug. 1982.
- [12] R.Y. Tsai and T.S. Huang, "Estimating Three-Dimensional Motion Parameters of a Rigid Planar Patch, III: Finite Point Correspondences and the Three View Problem," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-32, pp. 213-220, April 1984.
- [13] R.Y. Tsai and T.S. Huang, "Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects with Curved Surfaces," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-6, pp. 13-27, Jan. 1984.
- [14] J.K. Aggarwal and A. Mitiche, "Structure and Motion from Images: Fact and Fiction," in *Proc. Workshop on Computer Vision: Representation and Control*, Bellaire, MI, pp. 127-128, Oct. 1985.
- [15] J.Q. Fang and T.S. Huang, "Some Experiments on Estimating the 3-D Motion Parameters of a Rigid Body from Two Consecutive Image Frames," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-6, pp. 545-554, Sept. 1984.
- [16] T.S. Huang *et al.*, "Motion Detection and Estimation from Stereo Image Sequences: Some Preliminary Experimental Results," in *Proc. IEEE Workshop on Motion: Representation and Analysis*, Kiawah Island, SC, pp. 45-46, May 1986.
- [17] D.B. Gennery, "Tracking Known Three-Dimensional Objects," in *Proc. AAAI-82, National Conference on Artificial Intelligence*, Pittsburgh, PA, pp. 13-17, Aug. 1982.
- [18] D.B. Gennery, "Visual Tracking of Known Three-Dimensional Objects," *International Journal of Computer Vision*, vol. 7, pp. 243-270, 1992.
- [19] J. Weng, T.S. Huang, and N. Ahuja, "3-D Motion Estimation, Understanding, and Prediction from Noisy Image Sequences," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-9, pp. 370-389, May 1987.
- [20] L. Matthies and S.A. Shafer, "Error Modeling in Stereo Navigation," *IEEE Journal of Robotics and Automation*, vol. RA-3, pp. 239-248, June 1987.
- [21] E.D. Dickmanns and V. Graefe, "Dynamic Monocular Machine Vision," *Machine Vision and Applications*, vol. 1, pp. 233-240, 1988.
- [22] E.D. Dickmanns and V. Graefe, "Applications of Dynamic Monocular Machine Vision," *Machine Vision and Applications*, vol. 1, pp. 241-261, 1988.
- [23] H. Shariat and K. Price, "Motion Estimation with More Than Two Frames," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-12, pp. 417-434, May 1990.
- [24] C. Tomasi and T. Kanade, "Factoring Image Sequences into Shape and Motion," in *Proc. IEEE Workshop on Visual Motion*, Princeton, NJ, pp. 21-28, Oct. 1992.
- [25] Z. Zhang and O.D. Faugeras, "Three-Dimensional Motion Computation and Object Segmentation in a Long Sequence of Stereo Frames," *International Journal of Computer Vision*, vol. 7, pp. 211-241, 1992.
- [26] T.J. Broida and R. Chellappa, "Estimation of Object Motion Parameters from Noisy Images," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-8, pp. 90-99, Jan. 1986.
- [27] T.J. Broida and R. Chellappa, "Estimating the Kinematics and Structure of a Rigid Object from a Sequence of Monocular Images," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-13, pp. 497-513, June 1991.
- [28] T.J. Broida, S. Chandrashekar, and R. Chellappa, "Recursive Estimation of 3-D Kinematics and Structure from a Noisy Monocular Image Sequence," *IEEE Trans. Aerospace Electronic Systems*, vol. 26, pp. 639-656, July 1990.
- [29] S. Chandrashekar and R. Chellappa, "Passive Ranging Using a Moving Camera," *Journal of Robotic Systems*, vol. 9, pp. 729-752, Sept. 1992.
- [30] G.S. Young and R. Chellappa, "3-D Motion Estimation Using a Sequence of Noisy Stereo Images: Models, Estimation, and Uniqueness Results," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-12, pp. 735-759, Aug. 1990.
- [31] M. Porat and Y.A. Zeevi, "The Generalized Gabor Scheme of Image Representation in Biological and Machine Vision," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-10, pp. 452-468, July 1988.
- [32] Q. Zheng and R. Chellappa, "Automatic Feature Point Extraction and Tracking in Image Sequences for Arbitrary Camera Motion," Tech. Rep. CAR-TR-628, Center for Automation Research, University of Maryland, College Park, June 1992.
- [33] T.H. Wu and R. Chellappa, "Experiments on Estimating Motion and Structure Parameters Using Long Monocular Image Sequences," Tech. Rep. CAR-TR-640, Center for Automation Research, University of Maryland, College Park, Oct. 1992.
- [34] T.H. Wu and R. Chellappa, "Stereoscopic Recovery of Egomotion and Environmental Structure: Models, Uniqueness and Experiments," 1992. Submitted to *IEEE Conf. on Computer Vision and Pattern Recognition*, New York, NY, June 1993.
- [35] Q. Zheng and R. Chellappa, "A Computational Vision Approach to Image Registration," in *Proc. International Conf. on Pattern Recognition*, The Hague, The Netherlands, pp. 193-197, Aug. 1992.
- [36] R. Dutta *et al.*, "A Data Set for Quantitative Motion Analysis," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, San Diego, CA, pp. 159-164, June 1989.

Recognition and Tracking of 3D Objects by 1D Search

Daniel F. DeMenthon

Computer Vision Laboratory, Center for Automation Research,
University of Maryland, College Park, MD 20742-3275

Abstract

We show that the bounded-error recognition problem for images of 3D objects using point features can be decomposed into 1D search tasks, along lines joining the origin of the object coordinate system to the feature points chosen to model the object. Points are constructed along these lines at locations given by the coordinates of the detected image points; concurrent bracketing of these points by segment tree search along each of these lines provides maximal matchings between feature points and image points. Depth of search is limited by pixel resolution. This method is well adapted to the task of tracking objects in the presence of variable occlusion and clutter.

1 Introduction

The task considered here is model-based recognition and tracking using point features to describe 3D objects. We formulate the problem using the techniques introduced by Baird [1] and extended by Cass [4], Breuel [2, 3], and others [7, 8]. The approach consists of matching model features and image features to determine the pose of the object, while assuming that there are spurious or missing image features, and uncertainties in detecting these features. The image features are assumed to be located somewhere within bounded regions around their detected locations; the problem posed with this uncertainty model is sometimes called the *bounded-error* recognition problem. Authors have mostly applied this model to the matching of 2D images and the recognition of flat objects. One of the major obstacles to the practical extension of past work to general non-planar objects has been that the search in the general case has to be performed in an 8-dimensional transformation space [4]. The proposed method reduces this to *1D search by segment trees* along lines defined in the object model.

We introduce new equations for expressing the relationship between model points and image points in a perspective model of projection, generalizing a formulation introduced for iteratively computing object pose (the POSIT algorithm; see Appendix and [5]). These equations place the nonlinear terms of the transformation on the right hand side, in combination with the im-

age coordinate terms. The advantage of this formulation is that when initial estimates of these nonlinear terms are made, *the uncertainty in these estimates can be modeled as additional image uncertainty*. We obtain linear constraints on two 4D vectors I and J proportional to the first and second rows of the homogeneous transformation matrix of the object. These linear constraints represent *slabs* of space which are perpendicular to *feature vectors* (joining the origin of the object coordinate system to the object feature points) at points that depend on the image coordinates of these feature points. Regions of space where the largest numbers of slabs intersect correspond to maximal matchings between object points and image points. To find these regions we adapt the binary tree search advocated by Breuel [3] for this type of problem; in our formulation, the search can be decomposed into 1D searches by segment trees [9] along the feature vectors. Simultaneous searches are performed for the regions containing I and J , and are pruned by mutual constraints resulting from the fact that I and J belong to slabs corresponding to the same image points. Other pruning criteria use the fact that the first three components of I and J define vectors which are perpendicular and equal in length. When an object is tracked, the nonlinear terms of the equations can be evaluated from the pose results obtained for the previous image frame, and the dimensions of the initial search space can be reduced because the position and extent of this search space can be deduced from predictive techniques and bounds on admissible motions. This method accommodates the disappearance of features due to self-occlusion during the object's motion.

2 Notation

In Figure 1, we show the classic pinhole camera model, with its center of projection O , its image plane at distance f (the focal length) from O , its axes Ox and Oy pointing along the rows and columns of the camera sensor, and its third axis Oz pointing along the optical axis. The unit vectors for these three axes are called i , j and k . In this paper, the focal length and the intersection of the optical axis with the image plane (image center C) are assumed known.

An object with feature points $M_0, M_1, \dots, M_i, \dots, M_n$ is located in the field of view of the camera. The object coordinate frame is (M_0u, M_0v, M_0w) . The coordinates

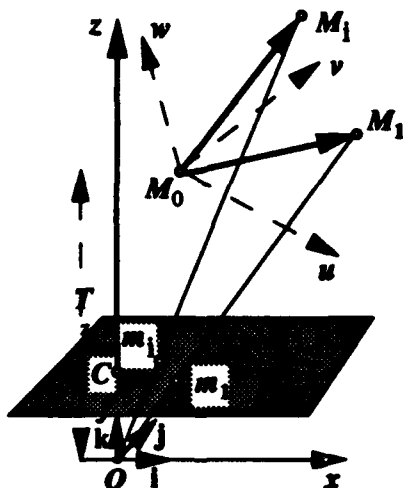


Figure 1: Perspective projections m_i for object points M_i

(U_i, V_i, W_i) of the points M_i in this frame are known. The images of the points M_i are called m_i , and the image coordinates (x_i, y_i) of each m_i are known. In the recognition problem, one of the goals is to be able to say that m_i is indeed the image of M_i , which is not obvious since the pose of the object is also unknown. In other words, the correspondences between the image points and the object points have to be found.

The rotation matrix R and translation vector T of the object in the camera coordinate system can be grouped into a single 4×4 transformation matrix which will be called the pose matrix P in what follows:

$$P = \begin{bmatrix} i_u & i_v & i_w & T_x \\ j_u & j_v & j_w & T_y \\ k_u & k_v & k_w & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

To obtain the coordinates of an object point M_i in the camera coordinate system using this pose matrix P instead of the more traditional rotation matrix and translation vector, one simply multiplies P by the coordinates of point M_i or vector M_0M_i in the object coordinate system. This operation requires that point M_i or vector M_0M_i be given a fourth coordinate (a fourth dimension) equal to 1. The four coordinates are said to be the homogeneous coordinates of the point or vector. In the following, vectors and points are four-dimensional (4D) entities in this homogeneous space, unless otherwise specified.

The first line of the matrix P is a row vector that we call P_1 . The other row vectors are called P_2 , P_3 and P_4 . The coordinates $T_x, T_y, T_z, 1$, the fourth column of the matrix, are the coordinates of the translation vector T (in Figure 1, the translation vector T is the vector OM_0). In the first row vector, P_1 , the coordinates i_u, i_v, i_w are the coordinates of a 3D vector, i , which is also the first row of the rotation matrix R of the transformation. Notice that i is also the unit vector for the x -axis of the camera coordinate system expressed in the object coordinate system (M_0u, M_0v, M_0w). Similarly,

in the second row vector, P_2 , the coordinates j_u, j_v, j_w are the coordinates of a 3D vector j which is the second row vector of the rotation matrix; j is also the unit vector for the y -axis of the camera coordinate system, expressed in the object coordinate system (M_0u, M_0v, M_0w). In the third row vector, P_3 , the coordinates k_u, k_v, k_w are the coordinates of a 3D vector k which is the cross product of i and j . In the following, we show how the vectors $I = \frac{f}{T_z}P_1, J = \frac{f}{T_z}P_2$ can be computed. Once they are obtained, T_z is found by noticing that the first three coordinates of I and J define 3D vectors R_1 and R_2 with norms equal to f/T_z . The completion of the object pose matrix P is then straightforward (see step 3 in the Appendix).

3 Fundamental Equations

The fundamental relations which relate the row vectors P_1, P_2 of the pose matrix, the coordinates of the object vectors M_0M_i in the object coordinate system, and the coordinates x_i and y_i of the perspective images m_i of M_i are

$$\begin{aligned} M_0M_i \cdot I &= x'_i, \\ M_0M_i \cdot J &= y'_i, \end{aligned} \quad (2)$$

with

$$I = \frac{f}{T_z}P_1, \quad J = \frac{f}{T_z}P_2, \quad (3)$$

$$x'_i = x_i(1 + \epsilon_i), \quad y'_i = y_i(1 + \epsilon_i), \quad (4)$$

and

$$\epsilon_i = M_0M_i \cdot P_3 / T_z - 1 \quad (5)$$

It is useful to introduce the unknown coordinates (X_i, Y_i, Z_i) of vector M_0M_i in the camera coordinate system for the sole purpose of demonstrating that these equations are correct. We remember that the dot product $M_0M_i \cdot P_1$ is the operation performed when multiplying the first row of the transformation matrix P by the coordinates of an object point in the object frame of reference to obtain the x -coordinate X_i of M_i in the camera coordinate system. Thus $M_0M_i \cdot P_1 = X_i$. For the same reason, the dot product $M_0M_i \cdot P_3$ is equal to Z_i , thus $(1 + \epsilon_i) = Z_i / T_z$. Also, in perspective projection, the relation $x_i = fX_i / Z_i$ holds between image point coordinates and object point coordinates in the camera coordinate system. Using these expressions in the equations above leads to identities, which proves the validity of the equations.

Two problems must be addressed before applying these equations:

1. The terms ϵ_i are generally unknown. These terms depend on P_3 (Equation (5)), which can be computed only after I and J have been computed (Section 2).
2. These equations can be used only after the correspondences between image points and object points have been established. Only then can the correct values for the image coordinate x_i be written on the right hand side for a given vector M_0M_i on the left hand side.

Starting with the first problem of dealing with unknown ε_i , notice that $x'_i = x_i(1 + \varepsilon_i)$ and $y'_i = y_i(1 + \varepsilon_i)$ are the image coordinates of the object points M_i when we use a scaled orthographic projection model. Indeed $x_i = fX_i/Z_i$ can be written as $x_i = \frac{f}{1+\varepsilon_i} X_i/T_z$, and we obtain $x'_i = fX_i/T_z$. In other words, image points (x'_i, y'_i) are obtained by "flattening" the object by orthographic projection of its points onto the plane $z = T_z$ through M_0 before performing a perspective projection. To obtain estimates for I and J , we first use x_i and y_i instead of x'_i and y'_i in Equations (2), thereby making errors $x_i\varepsilon_i$ and $y_i\varepsilon_i$ which are added to the estimates of the image errors. Once estimates for I and J have been obtained, these estimates can be used to find more precise values of ε_i , which in turn lead to better estimates of I and J .

Regarding problem (2), in some computer vision applications the correspondences can be obtained prior to the pose information. For example, in calibration applications, the feature points may be the centroids of marks that are easy to distinguish on the target calibration object. Then Equations (2) can be solved iteratively by first making rough estimates of the ε_i 's (setting them to zero when no information is available), solving the linear systems for I and J , finding better estimates for ε_i , and repeating the process. A least square object pose is generally found in a few iterations. The Appendix summarizes the steps of this algorithm, which was introduced in a more geometrical form in [5]. The rest of this paper addresses the more difficult situation where the correspondences between image points and object points cannot be obtained prior to the pose information.

4 Geometric Constraints for the Solutions I and J

The following discussion shows that the solutions I and J are located within small polyhedral regions which can be identified with respect to the 4D homogeneous coordinate system of the object.

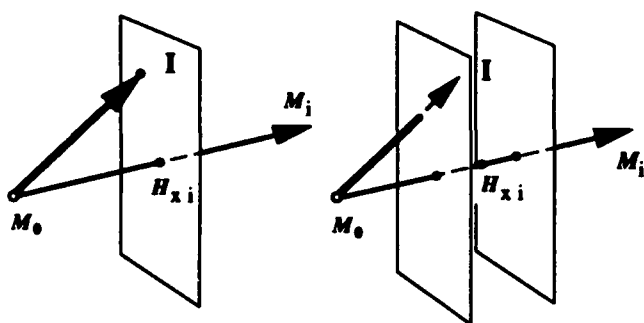


Figure 2: *Left:* In the absence of uncertainties, the head of vector I belongs to a plane orthogonal to the feature vector M_0M_i at the x -point H_{xi} located at abscissa $x'_i/|M_0M_i|$. *Right:* Because of uncertainties in image detection and ε_i , the head of I lies in a 4D slab perpendicular to M_0M_i .

Equations such as $M_0M_i \cdot I = x'_i$ (Equation (2)) can

be viewed as geometric constraints on the vector I in space with respect to the feature vectors M_0M_i : If the foot of vector I coincides with M_0 , the head of I must project on the feature line M_0M_i onto a point H_{xi} with abscissa $x'_i/|M_0M_i|$. Equivalently, the head of I must belong to a plane perpendicular to M_0M_i at H_{xi} (Figure 2). In the following, the points H_{xi} are called x -points. They are points constructed on the feature lines M_0M_i using the x -coordinates of the image points. Similarly, the points H_{yi} considered in constructing the vector J have abscissae $y'_i/|M_0M_i|$ and are called y -points.

In most situations, the terms $x'_i = x_i(1 + \varepsilon_i)$ are known only approximately. Bounds for the uncertainties in these terms can be computed by adding the image error e and the error e' made in estimating $x_i\varepsilon_i$. The ε_i terms are the projections of the vectors M_0M_i on the camera optical axis, divided by the distance T_z from the camera to the point M_0 along the camera optical axis (Equation (5)). Therefore an upper bound for these terms is R/D , where R is the radius of a sphere centered at M_0 containing the object and D is a lower bound for the distance T_z . Clearly, estimating tight upper bounds for these errors is made easier if we have some idea of the range in which the object is expected to be found. When the object is being tracked and an approximate pose for the object has been found from a previous image, ε_i can be estimated from this previous pose, and the uncertainty interval can be reduced and centered around $x_i(1 + \varepsilon_i)$.

Because of these uncertainties, all we can say is that the head of I projects onto the feature vector within the uncertainty interval around an x -point H_{xi} . Equivalently, the head of I must belong to a slab perpendicular to M_0M_i at x -point H_{xi} and with thickness defined by the uncertainty interval (Figure 2).

For I to be solution of a system of n Equations (2) for $i = 1, 2, 3, \dots, n$, the head of vector I must belong to the slab S_{11} defined at x -point H_{x1} on the feature vector M_0M_1 , the slab S_{22} defined at H_{x2} on M_0M_2 , etc. Therefore I must belong to the intersection of these n slabs. A necessary condition for this to occur is that there exists a region Σ in space contained in at least n slabs S_{ii} (Figure 3).

Similarly, J must belong to the intersection of n slabs T_{ii} . A necessary condition for this to occur is that there exist a region Θ in space contained in at least n slabs T_{ii} (Figure 3).

Furthermore, the n slabs S_{ii} that contain region Σ and the n slabs T_{ii} that contain region Θ must be defined by the same feature vectors M_0M_i and the same image points m_i . Therefore the n slabs T_{ii} at the y -points H_{yi} computed from the same points m_i which produced the slabs S_{ii} must intersect in a non-empty polyhedral region Θ .

As additional conditions, the solution vectors I and J are constrained in relative amplitude and orientation; the first three coordinates of I and J define two 3D vectors R_1 and R_2 . These vectors are proportional to i and j respectively, with the same coefficient of proportionality f/T_z . Therefore R_1 and R_2 must be orthogonal and have equal lengths. Therefore a pair of regions Σ and

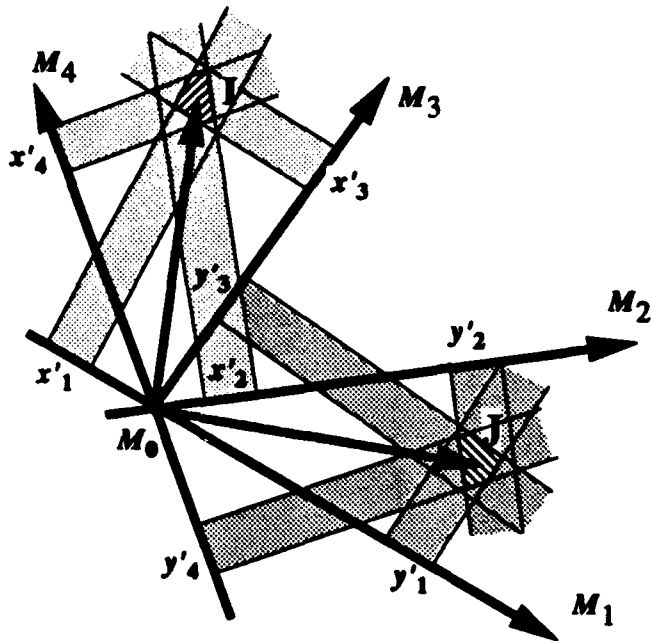


Figure 3: The head of I is found at the intersection of the slabs corresponding to the x -coordinates x'_i (corrected by $1 + \epsilon_i$) of the feature points M_i . The vector J is then found at the intersection of the slabs using the y -coordinates y'_i and the same correspondence. A further verification is obtained from the property that I and J (or 3D vectors defined by the first three coordinates of I and J in 4D) are perpendicular and have same lengths.

Θ can contain the heads of vectors I and J only if (1) the range of 3D distances from M_0 to the points of Σ overlaps the range of distances from M_0 to the points of Θ , and (2) the extrema of the 3D dot products of pairs of vectors with heads in each region have opposite signs.

5 Finding Solution Regions With Unknown Correspondences

In the problem addressed here, the correspondences between the N feature points and some of the n' detected image points are not known. Given an object point M_i , we do not know which image point among m_1, m_2, m_3 , etc., is the image of M_i . Furthermore, some points M_i may not have images, and some image points may not correspond to any of the object points. Let us assume for the moment that the number n_0 of image points that match the object points is known (finding this number is the objective of the next section).

The best we can then do is to consider, for the N feature points M_i , all the slabs defined by the n' detected image points. On each feature vector M_0M_i we can construct an x -point H_{xij} for each detected image point m_j , and consider the corresponding slab. Slabs obtained from a given feature vector M_0M_i are parallel, and slabs obtained from two different feature vectors intersect each other (object feature points M_i can be chosen so that the lines M_0M_i are well separated).

The proposed method finds small regions of space Σ

and Θ that contain the heads of vectors I and J . If indeed at least n_0 of the detected image points are images of object feature points, and if the bounds for the image and ϵ_i uncertainties are correct, there exists a pair of regions (Σ, Θ) such that Σ is contained in at least n_0 slabs, defined by x -points H_{xij} located on feature vectors M_0M_i and corresponding to image points m_j , and such that Θ is also contained in at least n_0 slabs, defined by y -points H_{yij} located on feature vectors M_0M_i and corresponding to the same image points m_j .

The method is based on eliminating pairs of regions of space which do not satisfy the geometric constraints defined in the previous section, proceeding from coarse to fine regions by bisection of space. A region Σ' and a region Θ' cannot contain the heads of I and J if:

1. Σ' or Θ' is not intersected by n_0 slabs (then no point inside the region can be contained in n_0 slabs).
2. Σ' and Θ' are not intersected by n_0 slabs constructed from the same image points.
3. The range of 3D distances from M_0 to the points of Σ' does not overlap the range of distances from M_0 to the points of Θ' . (Hence the two regions cannot respectively contain heads of I and J at equal distances from M_0).
4. The extrema of the 3D dot products of pairs of vectors with heads in each region have the same sign. (Hence the two regions cannot respectively contain heads of perpendicular vectors I and J).

There exists a pair of regions (Σ, Θ) which cannot be eliminated by the above criteria, and we can recognize and label in the image the n_0 points that contributed to these regions. We find these regions by simultaneously performing two recursive bisections of space. We simultaneously explore two binary trees by depth-first search, pairing branches of the two trees and pruning paired branches excluded by the above criteria.

To further verify the matching of the n_0 points (and also to provide a more accurate pose matrix), we proceed as follows: The terms ϵ_i can now be computed from the pose matrix, using Equation (5). The terms $x'_i = x_i(1 + \epsilon_i)$ and $y'_i = y_i(1 + \epsilon_i)$ can then be computed, as well as reduced uncertainty intervals. These corrected coordinates and intervals define thinner slabs at slightly different locations which result in smaller regions Σ and Θ , less ambiguity between possible matchings, and a more accurate pose.

6 Finding the Best Regions

The explanations so far have focused on finding regions Σ and Θ in the intersection of at least n_0 slabs. We would actually like to find regions contained in the intersection of the highest number of slabs, because this provides the maximal number of matches between image points and object points. We start the search with $n_0 = n'$, the total number of image points detected. Usually, some image points have no matches, and the search quickly fails. We then decrement n_0 until a search succeeds.

7 Searching for Regions Σ and Θ

A binary tree search was advocated by Breuel for this type of problem [3]. To search for a single region, say Σ , we start with a large box which is guaranteed to contain all the regions of interest, and recursively divide the box into two child boxes. At depth 1, the plane used to divide the box is perpendicular to the x -axis of the 4D space; at depth 2 the plane is perpendicular to the y -axis, at depth 3 to the z -axis, and at depth 4 to the k -axis. At depth 5 we are back to a division perpendicular to the x -axis, and so on. Eventually, we have divided the space into boxes so small that at least one of them is contained in the region Σ , in the intersection of n_0 slabs. This process is illustrated in 2D in Figure 4.

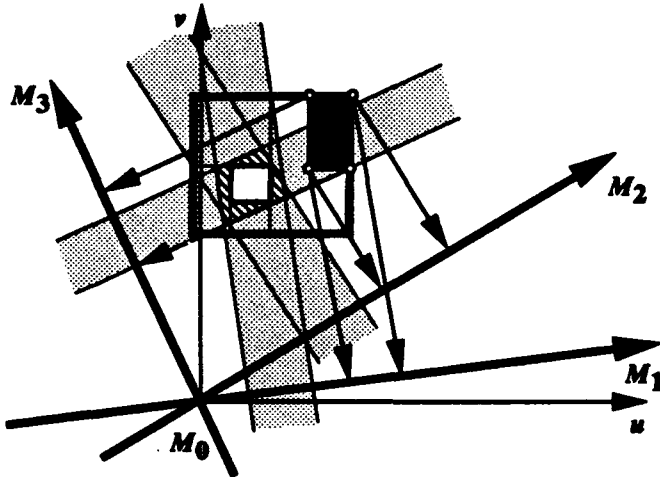


Figure 4: A search by bisection of space locates a box contained in a region in the intersection of three slabs (white box). Boxes which are not *intersected* by three slabs are pruned (black box). Tests for intersections between boxes and slabs are performed using box projections on the feature vectors.

7.1 Simultaneously Searching for Two Regions

We start with an initial box A_0 large enough to contain the head of I and an initial box B_0 large enough to contain the head of J (from Equations (2) and (3), one can show that the coordinates of these vectors can be expressed in pixels and are smaller than the largest image point coordinates). Box A_0 is divided into two boxes A_1 and A_2 , and box B_0 into two boxes B_1 and B_2 . The elimination criteria of the previous section are then applied to the pair of boxes A_1 and B_1 . If none of the criteria applies, the two boxes are themselves divided, and the process is repeated recursively. If any elimination criterion applies, another pair of boxes at the same depth, say A_1 and B_2 , is considered. If all four possible pairs are eliminated, we backtrack to a pair which was not considered at the previous depth. If the previous depth is the root depth, the search has failed. The search succeeds when two boxes of small predefined size (a few pixels, if the coordinates of I and J are expressed in pixels) survive the elimination criteria, and are contained in n_0 corresponding slabs; this second test is added when the boxes

become small enough to fit into the slabs; the elimination criteria use only the necessary (but not sufficient) condition that a box be intersected by n_0 slabs.

7.2 Tests for Intersection of a Box With n Slabs

If a box does not intersect n slabs, no subdivision of this box will be contained in n slabs, and this branch of the tree can be pruned [3]. This is one of the elimination criteria defined above. The tests for containment and intersection are simpler here than in Breuel's formulation. A box intersects n slabs if, for each of n feature vectors $\mathbf{M}_0\mathbf{M}_i$, the 1D projection of the box on $\mathbf{M}_0\mathbf{M}_i$ intersects the uncertainty interval around an x -point H_{xij} .

Instead of checking for intersections of intervals, we augment the interval of the box projection on each side by the amplitude of the uncertainty interval, and we count the x-points contained in this interval. The uncertainty intervals, and the lengths p_{d_i} of the projections of a box at depth d on the feature vectors $\mathbf{M}_0\mathbf{M}_i$, are precomputed (Figure 5).

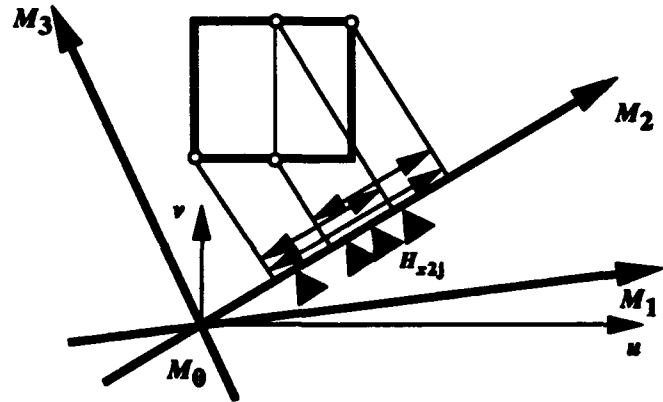


Figure 5: The projection segment of a child box shares one bound with the projection segment of its parent box. A binary search finds the other bound of this segment in the sorted parent list of x-points. This position provides the element count and the address for the child list.

The count of intersections between boxes and slabs is incremented by 1 for each feature vector when we find at least one x-point inside the (augmented) projection interval of the box. Thus we have to keep track of which x-points are contained in the augmented projection interval of the box. Having done this already for the parent box, we know the list of x-points contained in the parent interval. Each child interval shares one bound with the parent interval. The other bound is inside the parent interval and must be located with respect to the x-points (Figure 5). This is achieved by bisection of the parent list of x-points (the root list of x-points was sorted). The location of this bound provides the element count for the new list. The list of x-points for the left child has the same address as the parent list, whereas the list for the right child has an address offset by the position of its left bound in the parent list.

7.3 Tests for Containment of a Box in a Region

We verify that a box at depth d is contained in n_0 slabs by verifying that for each of at least n_0 feature vectors M_0M_i , the 1D projection of this box on M_0M_i is contained in the uncertainty interval around an x-point H_{xij} .

The depth for which all the lengths p_{di} of the projections of boxes at depth d on feature vectors M_0M_i are smaller than the lengths u_i of the uncertainty intervals is also precomputed, and we start checking for containment of the box projections in the uncertainty intervals only when the tree search has reached this depth. Instead of checking whether a projection is contained in an uncertainty interval around a point H_{xij} , we check whether there is a point H_{xij} in the interval of length $(u_i - p_{di})$.

8 Tracking

In a tracking task, we assume that the object has been found in the previous image frame, and its pose has been estimated after finding acceptable vectors I and J (Section 2). We perform the tracking in the 4D space where the search for I and J takes place. First, the previous pose allows us to compute estimates for the terms ϵ_i , and to take $x'_i = x_i(1 + \epsilon_i)$ and $y'_i = y_i(1 + \epsilon_i)$ as defining the positions of the x-points and y-points on the feature vectors. The error made by using ϵ_i from a previous frame contributes to the uncertainty intervals around these points and can be computed from upper bounds $d\theta$ and dT on possible rotation angle and translation increments between the two frames. Second, the vector I is transformed into $I' = I + dI$ between the two frames, and the search for the head of I' can be limited to a box centered around the head of I and of size larger than $|dI|$, also depending on $d\theta$ and dT . Predictive techniques can be used to predict I' and the uncertainty on I' in order to further reduce the size of the initial search box for I' , and similarly for J' .

9 Summary

We have introduced new equations that express the relationship between model points and image points in a perspective model of projection, the nonlinear terms of the perspective transformation are placed on the right hand side in combination with the image coordinate terms. The uncertainty in the estimates of these nonlinear terms can be modeled as additional image uncertainty. We obtain linear constraints on two 4D vectors I and J proportional to the first and second rows of the homogeneous transformation matrix of the object. These constraints represent slabs of space which are perpendicular to feature vectors (joining the origin of the object coordinate system to object feature points) at points that depend on the image coordinates of these feature points. Regions of space where the largest numbers of these slabs intersect are used to locate the vectors I and J and correspond to maximal matchings between object points and image points. Simultaneous binary tree searches are performed in regions containing I and J , and are pruned by mutual constraints resulting from the fact that I and J belong

to slabs corresponding to the same image points. Other pruning criteria use the fact that the first three components of I and J define vectors which are perpendicular and equal in length. Most of the search is 1D search by segment trees along the feature vectors.

Appendix: Iterative Pose Computation from Point Correspondences

Here we summarize a simple iterative algorithm for finding the pose of an object when a matching between object feature points and image points is known. It is an analytic formulation of the POSIT (Pose from Orthography and Scaling with Iterations) algorithm [5] in homogeneous form, which removes the need to locate the image of the origin M_0 of the object coordinate system. Note that this pose calculation is presented independently of the search method described above, which finds the matching and the pose by binary search of space when the matching is not known.

The equations to be solved are Equations (2). The steps of the iterative pose algorithm can be summarized as follows:

1. ϵ_i = best guess, or $\epsilon_i = 0$ if no pose information is available
2. Start of loop: Solve for I and J in the following systems (see next paragraph):

$$M_0M_i \cdot I = x'_i, M_0M_i \cdot J = y'_i$$

with

$$x'_i = x_i(1 + \epsilon_i), y'_i = y_i(1 + \epsilon_i)$$

3. From I , get

$$R_1 = (I_1, I_2, I_3),$$

$$f/T_z = |R_1|,$$

$$i = (T_z/f)R_1,$$

$$P_1 = (T_z/f)I$$

Similar operations yield j and P_2 from J .

4. $k = i \times j$, $P_3 = (k_u, k_v, k_w, T_z)$, $\epsilon_i = M_0M_i \cdot P_3/T_z - 1$
5. If all ϵ_i are close enough to the ϵ_i from the previous loop, EXIT, else go to step 2.
6. P_1, P_2, P_3 , along with $P_4 = (0, 0, 0, 1)$, are the four rows of the pose matrix.

We now provide details about finding I and J for step 2 of the iterative algorithm. The equations for I are

$$M_0M_i \cdot I = x'_i$$

The unknowns are the four coordinates (I_1, I_2, I_3, I_4) , of I , and we can write one equation for each of the object points M_i for which we know its position m_i in the image and its image coordinate x_i . One such equation has the form $U_i I_1 + V_i I_2 + W_i I_3 + I_4 = x'_i$, where $(U_i, V_i, W_i, 1)$ are the four coordinates of M_i . These equations for several object points M_i constitute a linear system of equations which can be written in matrix form as $AI = V_x$, where A is a matrix with i -th row vector $A_i = (U_i, V_i, W_i, 1)$, and V_x is a column vector with i -th coordinate x'_i .

Similarly, J can be found by solving the linear system $AJ = V_y$, where A is the same matrix, and V_y is a column vector with i -th coordinate y_i .

Since there are four unknown coordinates in vectors I and J , matrix A must have at least rank 4 for the systems to provide solutions. This requirement is satisfied if A has at least four rows and the object points are noncoplanar; therefore at least four noncoplanar object points and their corresponding image points are required. The pseudo-inversion operation is applied to matrix A ; the pseudo-inverse is called the object matrix B . Since A is defined in terms of the known coordinates of the object points in the object coordinate system, B depends only on the geometry of these object points and can be precomputed.

Experiments [5] show that this iterative approach generally provides an accurate pose of the object in a few iteration steps, as long as the points M_i are contained within a camera field of view of less than 90 degrees.

References

- [1] Baird, H.S., 1985, *Model-Based Image Matching Using Location*, MIT Press, Cambridge, MA.
- [2] Breuel, T.M., 1991, "Model Based Recognition using Pruned Correspondence Search", Proc. IEEE Conf. Computer Vision and Pattern Recognition, Maui, HI, pp. 257-262.
- [3] Breuel, T.M., 1992, "Fast Recognition using Adaptive Subdivisions of Transformation Space", Proc. IEEE Conf. Computer Vision and Pattern Recognition, Champaign, IL, pp. 445-451.
- [4] Cass, T.A., 1992, "Polynomial-Time Object Recognition in the Presence of Clutter, Occlusion, and Uncertainty", Computer Vision-ECCV 92, Lecture Notes in Computer Science 588, G. Sandini (Ed.), pp. 834-842, Springer-Verlag, Berlin.
- [5] DeMenthon, D.F., 1992, "Model-Based Object Pose in 25 Lines of Code", Proc. DARPA Image Understanding Workshop, San Diego, CA, pp. 753-761; also, Computer Vision-ECCV 92, Lecture Notes in Computer Science 588, G. Sandini (Ed.), pp. 335-343, Springer-Verlag, Berlin.
- [6] Grimson, W.E.L and D.P. Huttenlocher, 1988, "On the Sensitivity of the Hough Transform for Object Recognition", Proc. International Conf. Computer Vision, Tarpon Springs, FL.
- [7] Grimson, W.E.L, D.P. Huttenlocher, and D.W. Jacobs, 1991, "Affine Matching with Bounded Sensor Error: A Study of Geometric Hashing and Alignment", MIT AI Memo 1250.
- [8] Grimson, W.E.L, D.P. Huttenlocher, and D.W. Jacobs, 1992, "Affine Matching with Bounded Sensor Error", Computer Vision-ECCV 92, Lecture Notes in Computer Science 588, G. Sandini (Ed.), pp. 291-306, Springer-Verlag, Berlin.
- [9] Preparata, F.P., and M.I. Shamos, 1985, *Computational Geometry: An Introduction*, Springer-Verlag, Berlin.

Point Correspondence and Motion Detection in Long Image Sequences

Qinfen Zheng[†] and Rama Chellappa[‡]

[†]Department of Electrical Engineering

[‡]Center for Automation Research

University of Maryland

College Park, MD 20742-3275

Abstract

An automatic egomotion compensation based point correspondence and motion detection algorithm is presented. First, the motion of the camera is compensated using a computational vision based image registration algorithm. Then consecutive frames are transformed to the same coordinate system and the feature correspondence and motion detection problems are solved as though for a stationary camera. For point correspondence, feature points are detected using a Gabor wavelet decomposition and local interaction based algorithm. Methods of subpixel accuracy feature matching and tracking are introduced. For motion detection, we first determine the changed regions from the camera motion compensated frame differences. We then detect moving objects by grouping these changed regions and estimate the object motion parameters. Experimental results on several real image sequences are presented.

1 Introduction

A practical issue in camera pose estimation and structure from motion problems in computer vision research is the motion correspondence problem—automatic detection and tracking of features over successive frames. The basic task is to locate the same features over consecutive frames, a non-trivial problem when the camera motion between the frames is complicated, for example when there is significant camera rotation and translation between the frames and the camera motion is irregular. Tracking becomes even more difficult when dealing with automatically detected features since they may not always be located at significant points such as the corners of buildings. Various methods for solving the correspondence problem have been studied [3, 7]. In general, feature displacements over consecutive frames can be approximately decomposed into two components: (i) displacements due to camera motion that can be compensated by image rotation, scaling, and translation; (ii) displacements due to object motion and/or perspective deformation. The displacements due to camera motion are usually much larger and more irregular than the dis-

placements caused by object motion and perspective deformation. Most existing methods require the camera motion to be small and smooth. When tracking features in a long sequence, the problem of feature drift exists, i.e. the features may gradually change due to spatial sampling and to the deformation of the image due to the motions of objects and/or the camera. Feature drift alters the target being tracked and does not contain any useful information. Feature drift correction is especially important for tracking automatically detected features, which may be located in relatively smooth areas and hence may be more vulnerable to steady location drift.

In this paper, we introduce a two-step approach to solving the feature point correspondence problem. First, the motion of the camera is compensated using a computational vision based image registration algorithm [14]. A method for subpixel accuracy feature matching is then implemented to improve the camera motion compensation. Then consecutive frames are transformed to the same coordinate system and the feature point correspondence problem is posed as one of tracking moving objects using a still camera. A subpixel interpolation method is used to suppress feature drift.

Our approach is fully automatic and robust to various kinds of camera motion. It is simple because it compensates for camera motion at the first step, which significantly simplifies the matching process and reduces the computation. No higher level primitives such as edges, and no structure information, are required for the tracking step, and no post processing method such as relaxation [6] or Kalman filtering [5] is required. Good results have been obtained for several real image sequences acquired from indoor and outdoor scenes under different camera motions. Successful motion estimates based on the method of trajectory estimation reported here are described in [8, 11].

Automatic motion detection is an important practical problem. Intuitively, motion detection can be accomplished by taking differences of successive images and detecting non-zero parts. The problem becomes more complicated if the camera is on a moving platform so that translation, rotation, and scale change between the two images have to be taken into account. We introduce a two-step motion segmentation algorithm for detecting moving objects in image sequences acquired from a moving platform. First, the camera motion is compensated

using a subpixel accuracy image registration algorithm. We then select regions where changes may have occurred. A method of detecting actual moving objects and estimating their motion parameters is introduced.

In the following sections, we first present the camera motion estimation algorithm, then describe a subpixel-accuracy motion correspondence algorithm, and finally discuss the motion detection algorithm. Experimental results on motion correspondence and motion detection from several real sequences are presented at the end of the paper.

2 Camera Motion Estimation

2.1 Review of an Earlier Algorithm

Let (x_i, y_i) be the image frame coordinates, measured with respect to the position of the camera at time t_i , for $i = \{1, 2\}$. The relationship between the two frames can be approximated by [14]

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = s \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \begin{pmatrix} \Delta x_2 \\ \Delta y_2 \end{pmatrix} \quad (1)$$

where $s = \frac{\sqrt{\delta x_2^2 + \delta y_2^2}}{\sqrt{\delta x_1^2 + \delta y_1^2}}$ is the scaling factor, θ is the rotation angle between the two frames, and $(\Delta x_2, \Delta y_2)$ is the translation measured in the image coordinate system of frame t_2 . The effect of camera motion is characterized by the four parameters $\Delta x_2, \Delta y_2, \theta$, and s .

We use the image registration algorithm reported in [14] to estimate $\Delta x_2, \Delta y_2, \theta$, and s . The camera rotation is estimated and compensated early in the matching process using an illuminant direction estimator [12]. A small number of feature points are then located using a Gabor wavelet model for detecting local curvature discontinuities[4]. The feature points extracted from different frames are matched using a metric often used in area-based correlation techniques. Here, however, correlation is performed using feature points. Multiresolution transform-and-correct matching is implemented to obtain accurate estimates of camera motion parameters. At each resolution, frame t_1 is first transformed to the coordinates of frame t_2 using the estimated camera motion parameters $(\theta, \Delta x, \Delta y, s)$; then matching refinement is performed on the feature points of frame t_2 .

2.2 Subpixel Matching

The area correlation matching used here can only find the best grid point to grid point matches. Further processing is required for subpixel accuracy matching. Since a good initial match has been obtained by the area correlation step, a simple and effective way to achieve subpixel accuracy is by using an image differential method [1, 10]. Assume f_2 is offset by $(\delta x, \delta y)$ relative to f_1 ; then the frame difference can be written as

$$\begin{aligned} d(i, j) &= f_1(i, j) - f_2(i, j) \\ &\approx \frac{\partial f_1(i, j)}{\partial x} \delta x + \frac{\partial f_1(i, j)}{\partial y} \delta y \end{aligned} \quad (2)$$

where $\frac{\partial f_1}{\partial x}$ and $\frac{\partial f_1}{\partial y}$ are derivatives of f_1 and can be approximated by forward differences. For a small neighborhood around the feature point

we have $(2\omega_d + 1)^2$ simultaneous equations

$$-\omega_d \leq i, j \leq \omega_d$$

we have $(2\omega_d + 1)^2$ simultaneous equations

$$\vec{I} = G\vec{D} \quad (3)$$

The offset vector \vec{D} can be computed as

$$\vec{D} = \begin{pmatrix} \delta x \\ \delta y \end{pmatrix} = (G^T G)^{-1} G^T \vec{I} \quad (4)$$

where

$$\vec{I} = \begin{pmatrix} d(-\omega_d, -\omega_d) \\ \vdots \\ d(0, 0) \\ \vdots \\ d(\omega_d, \omega_d) \end{pmatrix}; \quad (5)$$

$$G = \begin{pmatrix} \frac{\partial f_1(-\omega_d, -\omega_d)}{\partial x} & \frac{\partial f_1(-\omega_d, -\omega_d)}{\partial y} \\ \vdots & \vdots \\ \frac{\partial f_1(0, 0)}{\partial x} & \frac{\partial f_1(0, 0)}{\partial y} \\ \vdots & \vdots \\ \frac{\partial f_1(\omega_d, \omega_d)}{\partial x} & \frac{\partial f_1(\omega_d, \omega_d)}{\partial y} \end{pmatrix} \quad (6)$$

In our implementation of this algorithm we use a small ω_d , for example $\omega_d = 4$, to reduce the computation and achieve better localization. Incorporation of subpixel accuracy feature matching significantly improves the accuracy of camera motion estimation. A test involving the estimation of simulated camera motion is presented in Section 5.

3 Motion Correspondence

3.1 Overview

As pointed out in Section 1, the fundamental task in motion correspondence is to track features over the image sequence. When the motion of the camera is compensated using the registration algorithm discussed in Section 2, the displacement of a feature point in the new frame can only be caused by perspective distortion and the motions of objects. In this paper we use intensity based area correlation to match features over consecutive frames. A problem associated with intensity based feature matching is that the locations of feature points are defined in terms of local intensity variations and may drift away after several frames. This can be caused by quantization of the feature locations and/or perspective deformation of the local image. In this paper, a method using subpixel accuracy tracking and weighted correlation is introduced to overcome this feature drift problem.

Consider the matching process illustrated in Figure 1. Although a feature point is located at grid point P_1 in the first frame, its location will be transformed to P'_1 to compensate for the camera motion. The exact location of P'_1 is usually not at a grid point. Since correlation-based matching can only locate the best matches at grid points, conventional approaches which approximate a feature location to its nearest grid point will result in two types of

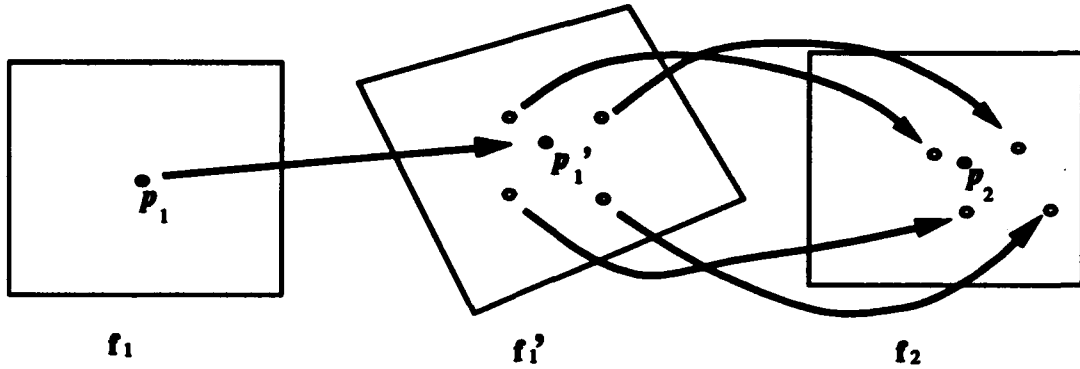


Figure 1: Feature matching graph

approximation errors, resulting from approximating P_1' and P_2 to their nearest grid points. Both errors cause the feature locations to migrate. The cumulative errors can be quite large when working with a long sequence. A subpixel accuracy tracking method is needed in order to obtain accurate trajectories. A three step subpixel accuracy matching algorithm is used here. First, an initial matching to the nearest grid point is achieved by a weighted cross correlation match. Secondly, the differential method discussed in Section 2.2 is used to achieve subpixel accuracy matching for all four nearest grid points. Finally, the feature location is interpolated from the matches of the four nearest grid points using an interpolation function.

The main steps in the algorithm are as follows. Given an image sequence, we first compute the camera motion parameters using the image registration algorithm presented in Section 2. Then, starting from the feature points detected in the first frame,¹ for every two consecutive frames we transform the first frame and the coordinates of its feature points to the coordinate system of the second frame. We then search the neighborhood of the anticipated feature locations for the best match. The locations of the feature points are then refined using the subpixel accuracy matching method discussed in Section 2.2 and an interpolation formula introduced in Section 3.3.

3.2 Weighted Correlation

As pointed out in Section 3.1, there are two sources of errors that cause feature drift: the error due to feature location quantization and the error due to image deformation caused by the relative motion between the camera and the objects. The feature location quantization error can be suppressed by using subpixel accuracy tracking. On the other hand, the error due to image deformation is usually more difficult to remove, especially when the area correlation method is used. In implementing the area correlation matching method, there is a trade-off in the

selection of the area size used in computing correlation; the larger the correlation area the better is the selectivity over similar features, but the less accurate are the feature locations. For example, when occlusion(s) occur within the correlation area, or when a feature point is near the border of an object and the motion of the camera changes the background significantly, the best cross-correlation will be away from the correct location, causing the feature locations to migrate. One way to solve this dilemma is to use a hierarchy of windows: use a large window to locate the correct matching peak, and gradually reduce the window size to better locate the feature points. We present a simple method using weighted correlation, in which greater weights are put on the neighbors which are closer to the feature center. The weighted correlation method possesses good selectivity since a large area is used, and at the same time it possesses good feature localization since the central parts have higher weights. The modified matching criterion is

$$\psi_{f_1 f_2} = \frac{1}{\sigma_1 \sigma_2} \sum_{ij} \gamma_{ij} \hat{f}_1(m+i, n+j) \hat{f}_2(u+i, v+j) \quad (7)$$

where

$$\begin{aligned} \hat{f}_1(m+i, n+j) &= f_1(m+i, n+j) - \mu_1 \\ \hat{f}_2(u+i, v+j) &= f_2(u+i, v+j) - \mu_2 \end{aligned}$$

$$\begin{aligned} \sigma_1 &= \sqrt{\sum_{ij} \gamma_{ij} \hat{f}_1^2(m+i, n+j)} \\ \sigma_2 &= \sqrt{\sum_{ij} \gamma_{ij} \hat{f}_2^2(u+i, v+j)} \\ \mu_1 &= \frac{1}{\Gamma} \sum_{ij} f_1(m+i, n+j) \\ \mu_2 &= \frac{1}{\Gamma} \sum_{ij} f_2(u+i, v+j) \\ \Gamma &= \sum_{ij} \gamma_{ij} \end{aligned} \quad (8)$$

The γ_{ij} s are non-negative weights. In our implementation, we let the γ_{ij} s be the same for pixels at the same

¹The feature points can also be obtained by other methods—for example, manually selected, as in many motion analysis algorithms. The motion correspondence part is independent of the selection of feature points.

distances from the center, where the distance is defined as the maximum distance along the x or y direction. To be more specific, we define the distance to be

$$k = d_{ij} = \max\{|i|, |j|\} \quad (9)$$

Notice that N_k , the total number of neighbors at the same distance k , is

$$N_k = \begin{cases} 1 & \text{for } k = 0 \\ 8k & \text{for } k \geq 1 \end{cases} \quad (10)$$

If we further require the sum of weights at each distance k to be constant (note that N_k increases as k increases, hence as a pixel moves farther away from the feature center its weight decreases), the weighting coefficient γ_{ij} can be generated using

$$\gamma_{ij} = \frac{\gamma_{00} c}{8(\max\{|i|, |j|\})}, \quad \{|i|, |j|\} \neq \{0, 0\} \quad (11)$$

where γ_{00} , the weight for the central point, is positive, for example $\gamma_{00} = 1$, and c is the ratio of γ_{00} to the weights of the outer layers. As shown in (11), the γ_{ij} s are positive and their values decrease as points move away from the central point; hence intensity similarities in the central part are given higher weights. Using Cauchy's inequality we can show [13] that the value of the weighted correlation (7) is in the range $[-1, 1]$, reaching the boundaries of the range if and only if

$$\frac{f_1(m+i, n+j) - \mu_1}{f_2(u+i, v+j) - \mu_2} = \text{const.} \quad \forall \{i, j\} \in \Omega \quad (12)$$

For example, when the two areas are identical we have

$$f_1(m+i, n+j) - \mu_1 = f_2(u+i, v+j) - \mu_2$$

and $\psi_{f_1, f_2} = 1$. On the other hand, when f_2 is the negative of f_1 we have

$$f_1(m+i, n+j) - \mu_1 = -[f_2(u+i, v+j) - \mu_2]$$

and $\psi_{f_1, f_2} = -1$.

3.3 Feature Location Interpolation

Using the differential method we can obtain subpixel accuracy matches for grid points. But for the feature tracking problem, after camera motion compensation, the features are usually not located at grid points. Consider the general feature matching situation illustrated in Figure 1. A feature point p_1 in frame t_1 is transformed to the coordinates of frame t_2 and located at $p'_1 = (x, y)$. Its four nearest grid points are (x_{11}, y_{11}) , (x_{12}, y_{12}) , (x_{21}, y_{21}) , and (x_{22}, y_{22}) determined by

$$\begin{aligned} x_{11} &= x_{21} = \text{int}(x) & y_{11} &= y_{12} = \text{int}(y) \\ x_{12} &= x_{22} = \text{int}(x) + 1 & y_{21} &= y_{22} = \text{int}(y) + 1 \end{aligned}$$

where $\text{int}(\cdot)$ is the truncation function.

Assuming that the matches to these points in frame t_2 are $(\hat{x}_{11}, \hat{y}_{11})$, $(\hat{x}_{12}, \hat{y}_{12})$, $(\hat{x}_{21}, \hat{y}_{21})$, and $(\hat{x}_{22}, \hat{y}_{22})$ respectively, we define the feature location (\hat{x}, \hat{y}) in frame t_2 to be

$$\begin{cases} \hat{x} = \hat{a}_1 x + \hat{a}_2 y + \hat{a}_3 xy + \hat{a}_4 \\ \hat{y} = \hat{b}_1 x + \hat{b}_2 y + \hat{b}_3 xy + \hat{b}_4 \end{cases} \quad (13)$$

where \hat{a}_i and \hat{b}_i , $i = \{1, 2, 3, 4\}$ are coefficients determined from the matches:

$$\begin{cases} \hat{x}_i = \hat{a}_1 x_i + \hat{a}_2 y_i + \hat{a}_3 x_i y_i + \hat{a}_4 \\ \hat{y}_i = \hat{b}_1 x_i + \hat{b}_2 y_i + \hat{b}_3 x_i y_i + \hat{b}_4 \end{cases} \quad (14)$$

where $i = \{11, 12, 21, 22\}$. The interpolation function (13) can be expressed relative to the match of (x_{11}, y_{11}) as

$$\begin{cases} \hat{X} - \hat{x}_{11} = a_1 \epsilon_x + a_2 \epsilon_y + a_3 \epsilon_x \epsilon_y + a_4 \\ \hat{Y} - \hat{y}_{11} = b_1 \epsilon_x + b_2 \epsilon_y + b_3 \epsilon_x \epsilon_y + b_4 \\ \epsilon_x = X - x_{11} \\ \epsilon_y = Y - y_{11} \end{cases} \quad (15)$$

Note that

$$\begin{aligned} x_{12} &= x_{11} + 1 & y_{12} &= y_{11} \\ x_{21} &= x_{11} & y_{21} &= y_{11} + 1 \\ x_{22} &= x_{11} + 1 & y_{22} &= y_{11} + 1 \end{aligned} \quad (16)$$

The coefficients (a_i s and b_i s) are [13]

$$\begin{aligned} a_1 &= \hat{x}_{12} - \hat{x}_{11} \\ b_1 &= \hat{y}_{12} - \hat{y}_{11} \\ a_2 &= \hat{x}_{21} - \hat{x}_{11} \\ b_2 &= \hat{y}_{21} - \hat{y}_{11} \\ a_3 &= \hat{x}_{22} + \hat{x}_{11} - \hat{x}_{12} - \hat{x}_{21} \\ b_3 &= \hat{y}_{22} + \hat{y}_{11} - \hat{y}_{12} - \hat{y}_{21} \\ a_4 &= 0 \\ b_4 &= 0 \end{aligned}$$

and the interpolation formula becomes

$$\begin{cases} \hat{X} = \hat{x}_{11} + (\hat{x}_{12} - \hat{x}_{11})\epsilon_x + (\hat{x}_{21} - \hat{x}_{11})\epsilon_y \\ \quad + (\hat{x}_{22} + \hat{x}_{11} - \hat{x}_{12} - \hat{x}_{21})\epsilon_x \epsilon_y \\ \hat{Y} = \hat{y}_{11} + (\hat{y}_{12} - \hat{y}_{11})\epsilon_x + (\hat{y}_{21} - \hat{y}_{11})\epsilon_y \\ \quad + (\hat{y}_{22} + \hat{y}_{11} - \hat{y}_{12} - \hat{y}_{21})\epsilon_x \epsilon_y \end{cases} \quad (17)$$

For convenience in further discussion, we note that

$$\begin{aligned} p_1 &= (x_{11}, y_{11}) & \hat{p}_1 &= (\hat{x}_{11}, \hat{y}_{11}) \\ p_2 &= (x_{12}, y_{12}) & \hat{p}_2 &= (\hat{x}_{12}, \hat{y}_{12}) \\ p_3 &= (x_{22}, y_{22}) & \hat{p}_3 &= (\hat{x}_{22}, \hat{y}_{22}) \\ p_4 &= (x_{21}, y_{21}) & \hat{p}_4 &= (\hat{x}_{21}, \hat{y}_{21}) \end{aligned} \quad (18)$$

Equation (17) can be considered as a generalization of the affine transform. When quadrangle $\hat{p}_1 \hat{p}_2 \hat{p}_3 \hat{p}_4$ is a parallelogram we have

$$\begin{cases} \hat{x}_{22} + \hat{x}_{11} - \hat{x}_{12} - \hat{x}_{21} = 0 \\ \hat{y}_{22} + \hat{y}_{11} - \hat{y}_{12} - \hat{y}_{21} = 0 \end{cases} \quad (19)$$

and (17) becomes

$$\begin{cases} \hat{x} = \hat{x}_{11} + (\hat{x}_{12} - \hat{x}_{11})\epsilon_x + (\hat{x}_{21} - \hat{x}_{11})\epsilon_y \\ \hat{y} = \hat{y}_{11} + (\hat{y}_{12} - \hat{y}_{11})\epsilon_x + (\hat{y}_{21} - \hat{y}_{11})\epsilon_y \end{cases} \quad (20)$$

which is the well-known affine transform. For more general quadrangles, the affine transform cannot guarantee correct matches at all four nearest grid points, while (17) does, as discussed below. Compared to the well-known second-order wrap transform

$$\begin{cases} \hat{x} = a_0 x^2 + a_1 y^2 + a_2 xy + a_3 x + a_4 y + a_5 \\ \hat{y} = b_0 x^2 + b_1 y^2 + b_2 xy + b_3 x + b_4 y + b_5 \end{cases} \quad (21)$$

(17) is simpler. Equation (21) has 12 coefficients to be determined, but only 4 matching pairs (8 measurements) are available. Hence (21) is an ill-posed problem, while (17) has only 8 coefficients and can be easily determined from the matches at the four corners. In addition to its simplicity, (17) possesses the nice properties discussed below:

Proposition 1 The four corners p_1, p_2, p_3 and p_4 are mapped into $\hat{p}_1, \hat{p}_2, \hat{p}_3$ and \hat{p}_4 , respectively.

Proof This is the constraint we use to define our mapping; hence it is always true. It can also be verified by substituting the coordinates of $p_i, i = \{1, 2, 3, 4\}$, into (17). \square

Proposition 2 Lines parallel to the x and y axes are mapped to lines in the transformed domain.

Proof For lines parallel to the x axis we have $\epsilon_y = \text{constant}$ and the interpolation functions can be written as

$$\begin{cases} \hat{x} = \hat{x}_{11} + a_1\epsilon_x + a_2\epsilon_y + a_3\epsilon_x\epsilon_y \\ \hat{y} = \hat{y}_{11} + b_1\epsilon_x + b_2\epsilon_y + b_3\epsilon_x\epsilon_y \end{cases} \quad (22)$$

which is the equation of a 2-D line written in parametric form.

Similarly, for lines parallel to the y axis we have $\epsilon_x = \text{constant}$ and the interpolation functions can be written as

$$\begin{cases} \hat{x} = \hat{x}_{11} + a_1\epsilon_x + a_2\epsilon_y + a_3\epsilon_x\epsilon_y \\ \hat{y} = \hat{y}_{11} + b_1\epsilon_x + b_2\epsilon_y + b_3\epsilon_x\epsilon_y \end{cases} \quad (23)$$

which again is the equation of a 2-D line written in parametric form. \square

Proposition 3 The four boundaries of the rectangle $\overline{p_1p_2p_3p_4}$ are mapped into the boundaries of the quadrangle $\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_4$.

Proof The boundaries of the rectangle $\overline{p_1p_2p_3p_4}$ are parallel to the axes, and from Proposition 2 we know they are mapped into line segments in the transformed domain. Also from Proposition 1 we have $\{p_i \rightarrow \hat{p}_i, i = 1, \dots, 4\}$. Since a line is uniquely determined by two points, and the transform functions in (17) are continuous, Proposition 3 follows. \square

Proposition 4 The bisectors of the rectangle $\overline{p_1p_2p_3p_4}$ are mapped into the bisectors of the quadrangle $\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_4$.

Proof Note that the bisectors of the rectangle $\overline{p_1p_2p_3p_4}$ are parallel to the axes, and from Proposition 2 their images are lines. To show that these lines are the bisectors of the quadrangle $\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_4$ we only need to prove that they bisect the opposite sides. First consider the bisector parallel to the X axis:

$$\begin{cases} x = x_{11} + \epsilon_x & 0 \leq \epsilon_x \leq 1 \\ y = y_{11} + 1/2. \end{cases} \quad (24)$$

Its image in the transform domain is

$$\begin{cases} \hat{x} = \hat{x}_{11} + \frac{1}{2}a_2 + (a_1 + \frac{1}{2}a_3)\epsilon_x \\ \hat{y} = \hat{y}_{11} + \frac{1}{2}b_2 + (b_1 + \frac{1}{2}b_3)\epsilon_x. \end{cases} \quad (25)$$

When $\epsilon_x = 0$ we have

$$\begin{cases} \hat{x} = \hat{x}_{11} + \frac{1}{2}a_2 = \frac{1}{2}(\hat{x}_{11} + \hat{x}_{21}) \\ \hat{y} = \hat{y}_{11} + \frac{1}{2}b_2 = \frac{1}{2}(\hat{y}_{11} + \hat{y}_{21}) \end{cases} \quad (26)$$

which is the center of p_1 and p_4 . Similarly at the other end we have $\epsilon_x = 1$ and

$$\begin{cases} \hat{x} = \hat{x}_{11} + \frac{1}{2}a_2 + (a_1 + \frac{1}{2}a_3) = \frac{1}{2}(\hat{x}_{12} + \hat{x}_{22}) \\ \hat{y} = \hat{y}_{11} + \frac{1}{2}b_2 + (b_1 + \frac{1}{2}b_3) = \frac{1}{2}(\hat{y}_{12} + \hat{y}_{22}) \end{cases} \quad (27)$$

which is the center of p_2 and p_3 . So (25) bisects two opposite sides of the quadrangle $\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_4$. Similarly we can show the bisector parallel to the Y axis bisects the other two opposite sides of the quadrangle $\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_4$. \square

Proposition 5 The center of the rectangle $\overline{p_1p_2p_3p_4}$ is mapped into the intersection of the bisectors of the quadrangle $\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_4$, which is the center of the corners of the quadrangle $\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_4$.

Proof The center of the rectangle $\overline{p_1p_2p_3p_4}$ is at the intersection of the two bisectors. From Proposition 4, its image must be on the bisectors of the quadrangle $\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_4$, i.e. the intersection of the bisectors of the quadrangle $\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_4$.

The center of the rectangle $\overline{p_1p_2p_3p_4}$ is $(x_{11} + \frac{1}{2}, y_{11} + \frac{1}{2})$. Its image in frame f_2 is

$$\begin{cases} \hat{x} = \hat{x}_{11} + \frac{1}{2}(\hat{x}_{12} - \hat{x}_{11}) + \frac{1}{2}(\hat{x}_{21} - \hat{x}_{11}) \\ \quad + \frac{1}{4}(\hat{x}_{22} + \hat{x}_{11} - \hat{x}_{12} - \hat{x}_{21}) \\ \quad = \frac{1}{4}(\hat{x}_{11} + \hat{x}_{12} + \hat{x}_{21} + \hat{x}_{22}) \\ \hat{y} = \hat{y}_{11} + \frac{1}{2}(\hat{y}_{12} - \hat{y}_{11}) + \frac{1}{2}(\hat{y}_{21} - \hat{y}_{11}) \\ \quad + \frac{1}{4}(\hat{y}_{22} + \hat{y}_{11} - \hat{y}_{12} - \hat{y}_{21}) \\ \quad = \frac{1}{4}(\hat{y}_{11} + \hat{y}_{12} + \hat{y}_{21} + \hat{y}_{22}) \end{cases} \quad (28)$$

Hence

$$(\hat{x}, \hat{y}) = \frac{1}{4}(\hat{p}_1 + \hat{p}_2 + \hat{p}_3 + \hat{p}_4) \quad (29)$$

This shows that the center of the rectangle $\overline{p_1p_2p_3p_4}$ is mapped into the centroid of the corners of the quadrangle $\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_4$. \square

Proposition 6 For a point inside the rectangle $\overline{p_1p_2p_3p_4}$, its image is also inside the quadrangle $\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_4$ provided the quadrangle $\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_4$ is convex.

Proof Any point inside the rectangle $\overline{p_1p_2p_3p_4}$ in frame f_1 can be expressed as

$$\begin{cases} p = (x_{11} + \delta x, y_{11} + \delta y) \\ 0 \leq \delta x \leq 1 \\ 0 \leq \delta y \leq 1 \end{cases} \quad (30)$$

Its image in frame f_2 can be computed as [13]

$$\hat{p} = (1 - \delta y)[(1 - \delta x)\hat{p}_1 + \delta x\hat{p}_2] + \delta y[(1 - \delta x)\hat{p}_4 + \delta x\hat{p}_3] \quad (31)$$

From Proposition 3 we know that $\hat{p} = (1 - \delta x)\hat{p}_1 + \delta x\hat{p}_2$ is on boundary $\overline{\hat{p}_1\hat{p}_2}$ and $\hat{p} = (1 - \delta x)\hat{p}_4 + \delta x\hat{p}_3$ is on boundary $\overline{\hat{p}_4\hat{p}_3}$. Hence the linear combination $\hat{p} = (1 - \delta y)\hat{p} + \delta y\hat{p}$ is inside the quadrangle $\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_4$ when the quadrangle $\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_4$ is convex. \square

Proposition 7 *The convexity of a quadrangle $\overline{a_1 a_2 a_3 a_4}$ can be verified by checking the areas of the triangles $\Delta \overline{a_1 a_2 a_3}$, $\Delta \overline{a_1 a_4 a_3}$, $\Delta \overline{a_2 a_3 a_4}$, and $\Delta \overline{a_2 a_1 a_4}$. The quadrangle $\overline{a_1 a_2 a_3 a_4}$ is convex if and only if*

$$S_{\overline{a_1 a_2 a_3}} + S_{\overline{a_1 a_4 a_3}} = S_{\overline{a_2 a_3 a_4}} + S_{\overline{a_2 a_1 a_4}} \quad (32)$$

Proof Four different triangles can be formed from the corners of the quadrangle. If the quadrangle is convex, then the sum of the areas of the complementary triangles equals the area of the quadrangle. Hence equality holds for (32). On the other hand, when the quadrangle is concave, the sum of one pair of complementary triangles still equals the area of the quadrangle. But the other two complementary triangles contain areas outside the quadrangle. Hence the sum of their areas is larger than the area of the quadrangle and the two sides of (32) are not equal.

This is a simple way to check the convexity of a quadrangle. The area of a triangle $\Delta \overline{a_1 a_2 a_3}$ can be calculated as

$$S_{\Delta \overline{a_1 a_2 a_3}} = \pm \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \quad (33)$$

where $x_i, y_i, i = \{1, 2, 3\}$ are the coordinates of $a_i, i = \{1, 2, 3\}$. \square

3.4 Displacement Prediction

The compensation for global rotation, scale, and translation between the two frames makes the displacement of feature locations between consecutive frames more predictable. Accordingly, the search window for matching can be decreased to reduce computations and mismatches. Nevertheless, since the amount of displacement is inversely proportional to the depth of the object along the optical axis and proportional to the distance between the feature point and the focus of expansion in the image domain, for an object moving close to the camera, the displacements of corresponding features can be quite large. Fortunately, when the motion of the camera is compensated first, feature displacements can only be translations, due to the motions of objects, and "pure" perspective distortions due to changes in the depths of the objects. Hence the displacements of the features in the new frame can be predicted using results from previous frames. We have used a zeroth order predictor; the displacement of a feature in the two preceding frames is used as the bias in tracking the feature in the current frame.

4 Motion Detection

4.1 Change Detection

With the camera motion parameters estimated using the procedure discussed in Section 2, we transform the images into a common coordinate system and take the difference between successive frames. We then detect changed parts from the camera motion compensated frame difference. Two factors need to be addressed. First, since images always contain noise, an intensity change between frames can be due to either object motion or image noise. A difference between intensity

changes due to object motion and changes due to image noise is that changes due to object motion usually occur in conjunction with changes in the intensities of nearby pixels, while changes due to image noise are usually isolated. Thus by checking the values of both the average of the local intensity changes and the on-site intensity change, we can suppress image noise and detect changes due to object motion. Secondly, since the value of the intensity change is dependent on the local contrast of the image, in high-contrast areas intensity interpolation due to camera motion compensation can cause intensity changes. Hence the threshold used for segmenting moving parts should be adjusted according to the local image contrast.

In this paper we classify a pixel (i, j) as belonging to a changing part if

$$\min\{\overline{d_{i,j}}, d_{i,j}\} > \max\{\gamma D_{i,j}, T\} \quad (34)$$

where

$$\overline{d_{i,j}} = \frac{1}{N_\omega} \sum_{i,j \in \omega} d_{i,j}$$

$$D_{i,j} = \sup_{i,j \in \omega} \{f_{i,j}\} - \inf_{i,j \in \omega} \{f_{i,j}\}$$

$$d_{i,j} = |f'_{i,j} - f_{i,j}|$$

$f'_{i,j}$: value in the first frame

$f_{i,j}$: value in the second frame

γ : Threshold for relative difference

T : Threshold for absolute difference

ω : The neighborhood clique

In our implementation, we took ω to be the 3×3 neighborhood.

4.2 Motion Estimation

When more than two successive frames are available, we can segment moving objects from the background and estimate their velocities.

Let Ω be the set of pixels which belong to a moving object, and let Ω_1, Ω_2 and Ω_3 be these sets of pixels in the camera motion compensated frames t_1, t_2 and t_3 respectively. Using the method discussed in Section 4.1 we can detect $S_1 = \Omega_1 \cup \Omega_2$ from the difference of frames t_1 and t_2 and $S_2 = \Omega_2 \cup \Omega_3$ from the difference of frames t_2 and t_3 . In general, due to the similarity between pixels belonging to the same object, the virtually detected changed parts are S'_i , which is a subset of S_i , for $i = 1, 2$. Assuming that the displacement from Ω_1 to Ω_2 is (u_1, v_1) , we have

$$S'_1 = S_1 - S_1^o \quad (35)$$

where

$$S_1^o = \{(i, j) | (i, j) \in \Omega_1 \cap \Omega_2, f_1(i, j) = f_2(i - u_1, j - v_1)\}$$

Similarly, assuming that the displacement from Ω_2 to Ω_3 is (u_2, v_2) we have

$$S'_2 = S_2 - S_2^o \quad (36)$$

where

$$S_2^o = \{(i, j) | (i, j) \in \Omega_2 \cap \Omega_3, f_2(i, j) = f_3(i - u_2, j - v_2)\}$$

Note that

$$(i, j) \in \Omega_1 \Leftrightarrow (i+u_1, j+v_1) \in \Omega_2 \\ \Leftrightarrow (i+u_1+u_2, j+v_1+v_2) \in \Omega_3 \quad (37)$$

So

$$\Omega_2 = \Omega_1 \oplus (u_1, v_1) \\ \Omega_3 = \Omega_2 \oplus (u_2, v_2) = \Omega_1 \oplus (u_1 + u_2, v_1 + v_2)$$

where \oplus adds the coordinates of each pixel in the left operand to those in the right operand.

When $u_1 = u_2 = u$ and $v_1 = v_2 = v$ we have

$$(i, j) \in S'_1 \Leftrightarrow (i+u, j+v) \in S'_2$$

so that

$$S'_2 = S'_1 \oplus (u, v) \quad (38)$$

Let (x_i, y_i) be the mass center of S'_i ; then

$$x_1 = \frac{1}{N_{S_1}} \sum_{(i,j) \in S'_1} i \quad (39)$$

$$y_1 = \frac{1}{N_{S_1}} \sum_{(i,j) \in S'_1} j \quad (40)$$

$$x_2 = \frac{1}{N_{S_2}} \sum_{(i,j) \in S'_2} i = \frac{1}{N_{S_1}} \sum_{(i,j) \in S'_1} i+u = x_1+u \quad (41)$$

$$y_2 = \frac{1}{N_{S_2}} \sum_{(i,j) \in S'_2} j = \frac{1}{N_{S_1}} \sum_{(i,j) \in S'_1} j+v = y_1+v \quad (42)$$

where N_{S_i} is the number of pixels in set S'_i for $i = 1, 2$. The motion velocity (u, v) can then be estimated from the change in the mass center of the detected changed area.

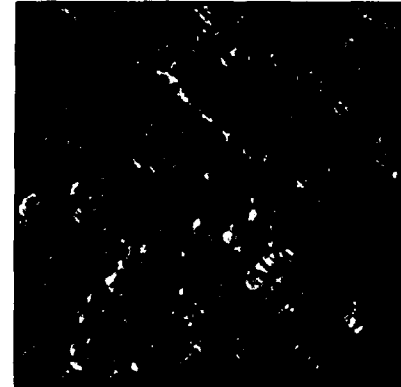
We then refine the velocity estimate by shifting the first frame by (u, v) and computing the subpixel accuracy match using (4). While doing this only pixels belong to S'_1 are used to get \tilde{I} and G in (5) and (6). We next construct a candidate set for $\tilde{\Omega}_1$, $\tilde{\Omega}_1$, as the closure of S'_1 . $\tilde{\Omega}_1$ is recursively generated from S'_1 by connecting any two points which are inside $\tilde{\Omega}_1$ and have the same value for one of their coordinates. Finally, we detect Ω_1 by translating frame f_1 by the estimated (u, v) (after subpixel accuracy refinement) to get f'_1 , and detect zeros of $|f'_1 - f_2|$. A pixel in $\tilde{\Omega}_1$ is classified as belonging to Ω_1 , the moving object, if $|f'_{1,i,j} - f_{2,i,j}| \leq \epsilon$ and at least one of the following conditions is satisfied: (1) $(i, j) \in S'_1$; or (2) at least one of its 3×3 neighbors belongs to Ω_1 , where ϵ is a threshold for detecting the zeros in frame difference.

5 Experiments

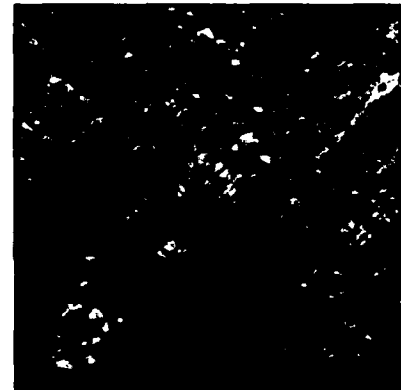
5.1 Motion Correspondence

Our first example shows the improvements in camera motion estimation that result from using the subpixel accuracy feature matching technique discussed in Section 2.2. Figure 2 shows a pair of balloon images with synthetic camera motion [14]. The true camera motion parameters are $\theta = 90^\circ$, $\delta x = 50$, $\delta y = 50$, and $s = 0.9$.

In [14] it was reported that the estimated parameters were $\theta = 89.8083^\circ$, $\delta x = 49.907$, $\delta y = 50.047$, and $s = 0.90138$. For the same set of images, camera motion parameters equal to $\theta = 90.000^\circ$, $\delta x = 50.0010$, $\delta y = 49.9997$, and $s = 0.900018$ have been obtained by using the improved image registration algorithm discussed in Section 2.2. The improvements are significant.



(a) B036



(b) Transformed B036

Figure 2: Images with synthetic camera motion.

Our first tracking example is the PUMA2 sequence [2]. The sequence consists of thirty 256×256 images.² The camera motion is a continuous rotation. For discussion purposes, in this example we give the tracking results on twenty manually selected points.³ Figure 3.a shows the selected feature points on the first frame. Figures 3.b and 3.c show their trajectories tracked up to frames 15 and 30 before the subpixel accuracy tracking technique was implemented. Figures 3.d and 3.e show the trajectories tracked up to frames 15 and 30 after the subpixel accuracy tracking technique was used. Figure 3.f shows the feature points hand-picked from frame 30. Note that without enforcing subpixel accuracy tracking, most of the trajectories gradually drift away from the initial chosen points. Comparison of Figures 3.c, 3.e and 3.f shows that the error in feature drift has been

²There are black strips on the tops of the original images. In Figure 3 these black strips have been removed, i.e. only the bottom 256×242 is displayed.

³Results of tracking another set of manually selected points and a set of automatically detected feature points are given in [13].

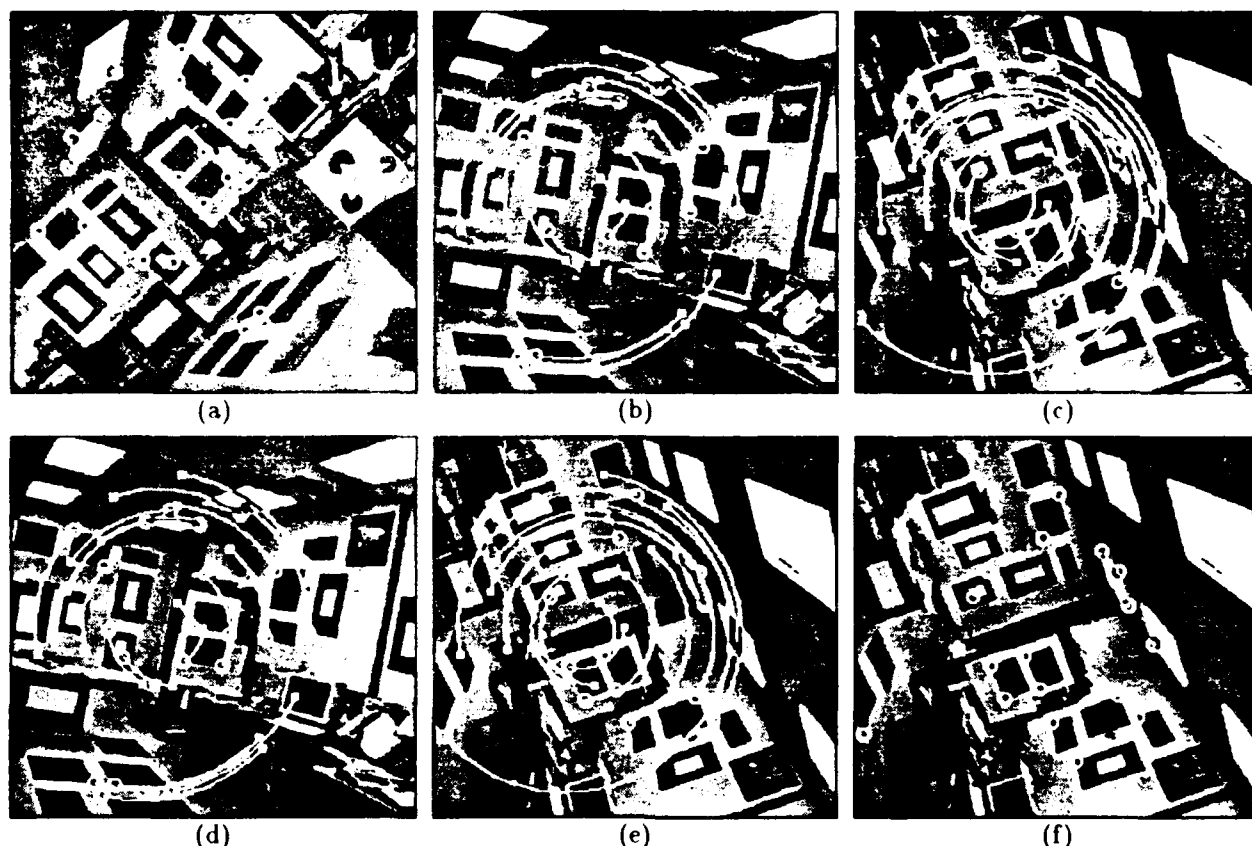


Figure 3: Motion correspondence for PUMA2 sequence. (a) Feature points are manually picked in the first frame. (b) Trajectories of the feature points tracked up to the 15th frame without using the subpixel accuracy tracking technique. (c) Trajectories of the feature points tracked up to the 30th frame without using the subpixel accuracy tracking technique. (d) Trajectories of the feature points tracked up to the 15th frame after using the subpixel accuracy tracking technique. (e) Trajectories of the feature points tracked up to the 30th frame after using the subpixel accuracy tracking technique. (f) The same feature points manually picked in the 30th frame.

significantly reduced. Table 1 lists the coordinates of the feature points obtained by our algorithm as well as those selected manually. The differences in the feature coordinates are within the limits of accuracy for manual selection.

Figure 4 shows an example using an outdoor image sequence. The original sequence consists of thirty 512×512 images. In this sequence, the motion of the camera is not smooth. Also there are rapidly moving clouds causing significant illuminant changes. In our experiments we first cut 36 lines from the top and bottom of each frame to remove the black strips on the bottoms of the original images, so the sequence we are working with consists of the central 512×440 subimages of the original sequence. Figure 4.a shows feature points automatically detected from the first frame. Figures 4.b and 4.c show their trajectories tracked up to frames 15 and 30 respectively. As shown in Figure 4, the trajectories are ragged due to non-smooth camera motion. Nevertheless, the algorithm tracks the same features.

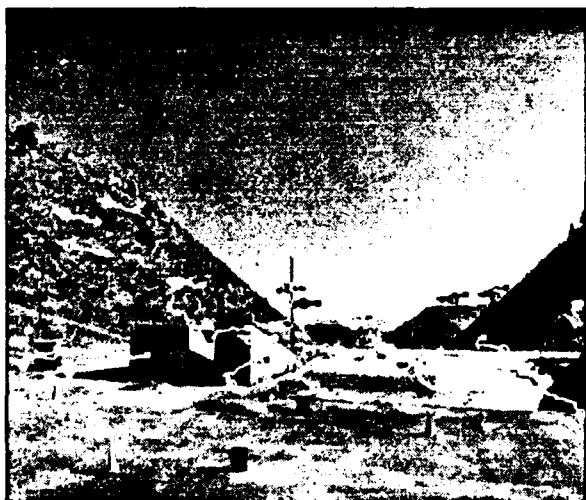
Our third tracking example is a sequence taken from a camera on a helicopter which was flying along a straight line path between rows of trucks parked on the runway [9]. The original sequence consists of ten 512×512 im-

Table 1: Feature locations on frames 1 and 30

feature No.	Manually selected				Tracking results	
	frame 1		frame 30		frame 30	
	X	Y	X	Y	X	Y
1	38	94	167	100	168.82	100.36
2	57	81	173	122	172.01	120.98
3	36	113	156	87	156.07	90.25
4	65	53	188	146	189.56	148.73
5	47	150	119	79	117.77	79.17
6	17	153	129	51	128.82	51.73
7	83	172	84	100	83.84	101.54
8	102	173	75	116	73.74	117.38
9	148	216	—	—	—	—
10	179	180	6	204	7.17	205.39
11	157	206	—	—	—	—
12	159	213	—	—	—	—
13	105	58	166	179	166.08	180.39
14	148	59	140	221	139.48	221.55
15	138	86	125	196	124.00	196.87
16	108	39	179	193	178.18	193.62
17	122	103	117	175	116.12	175.85
18	105	120	107	154	107.65	153.44
19	137	119	95	181	95.64	181.72
20	121	36	86	160	86.17	160.60



(a)

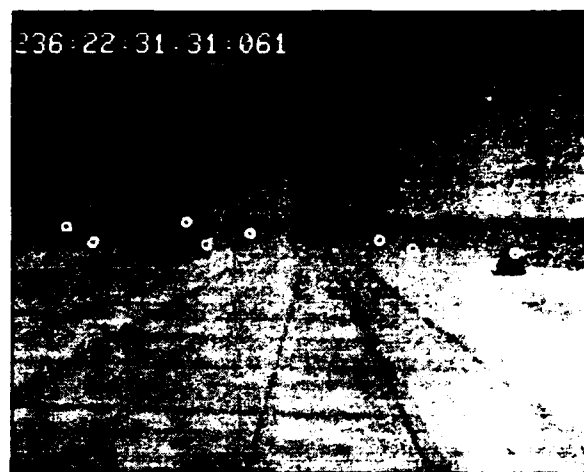


(b)

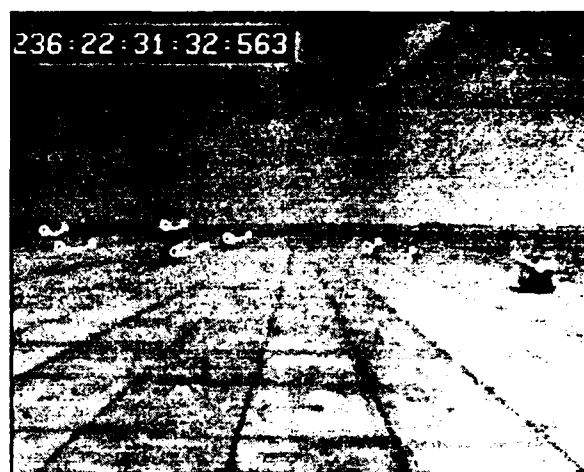


(c)

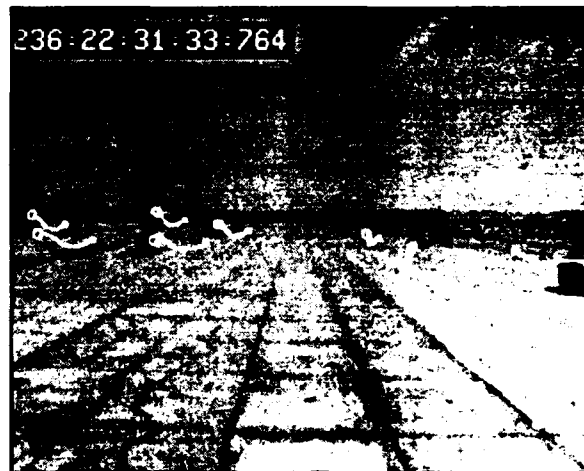
Figure 4: Motion correspondence for the rocket sequence. (a) Feature points are automatically detected in the first frame. (b) Trajectories of the feature points tracked up to the 15th frame. (c) Trajectories of the feature points tracked up to the 30th frame.



(a)



(b)



(c)

Figure 5: Motion correspondence for NASA helicopter line sequence. (a) Feature points are automatically detected in the first frame. (b) Trajectories of the feature points tracked up to the 6th frame. (c) Trajectories of the feature points tracked up to the 10th frame.

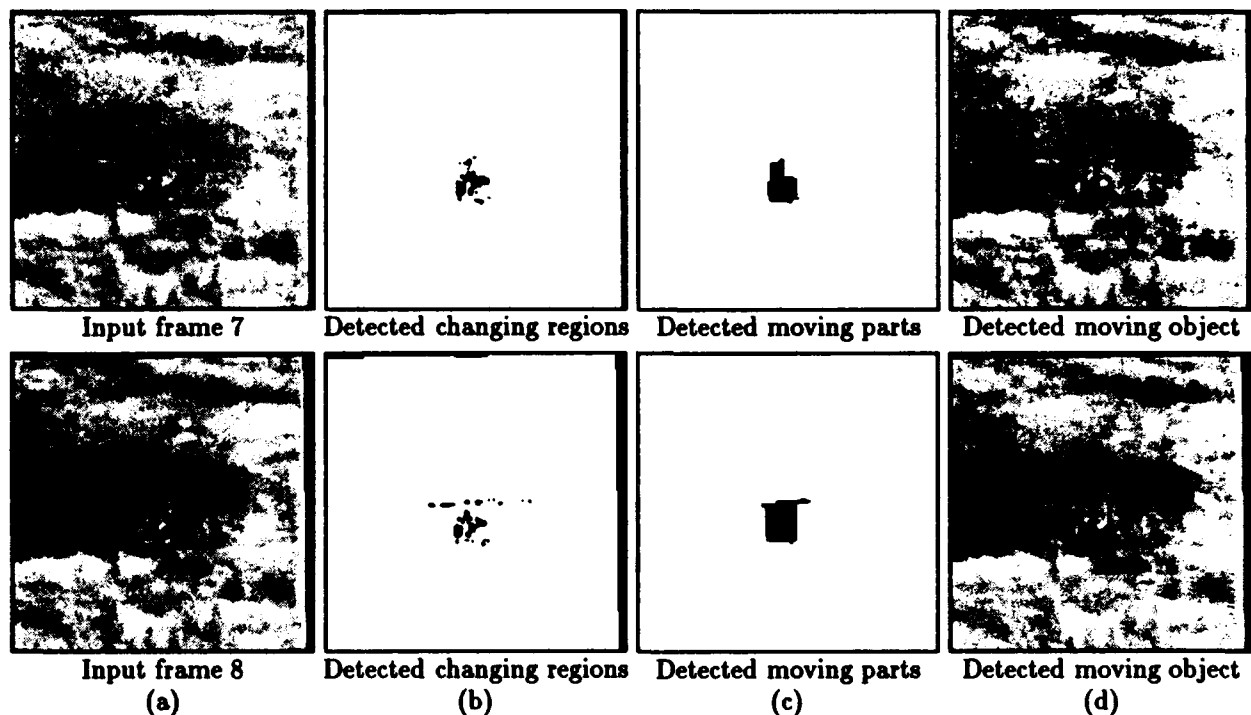


Figure 6: Moving object detection from a helicopter sequence. (a) Input frames. (b) Changing regions segmented from camera motion compensated frame differences. (c) Moving part detected using our algorithm. (d) Superimposition of the contour of the detected moving part on the image to show the correctness of motion detection.

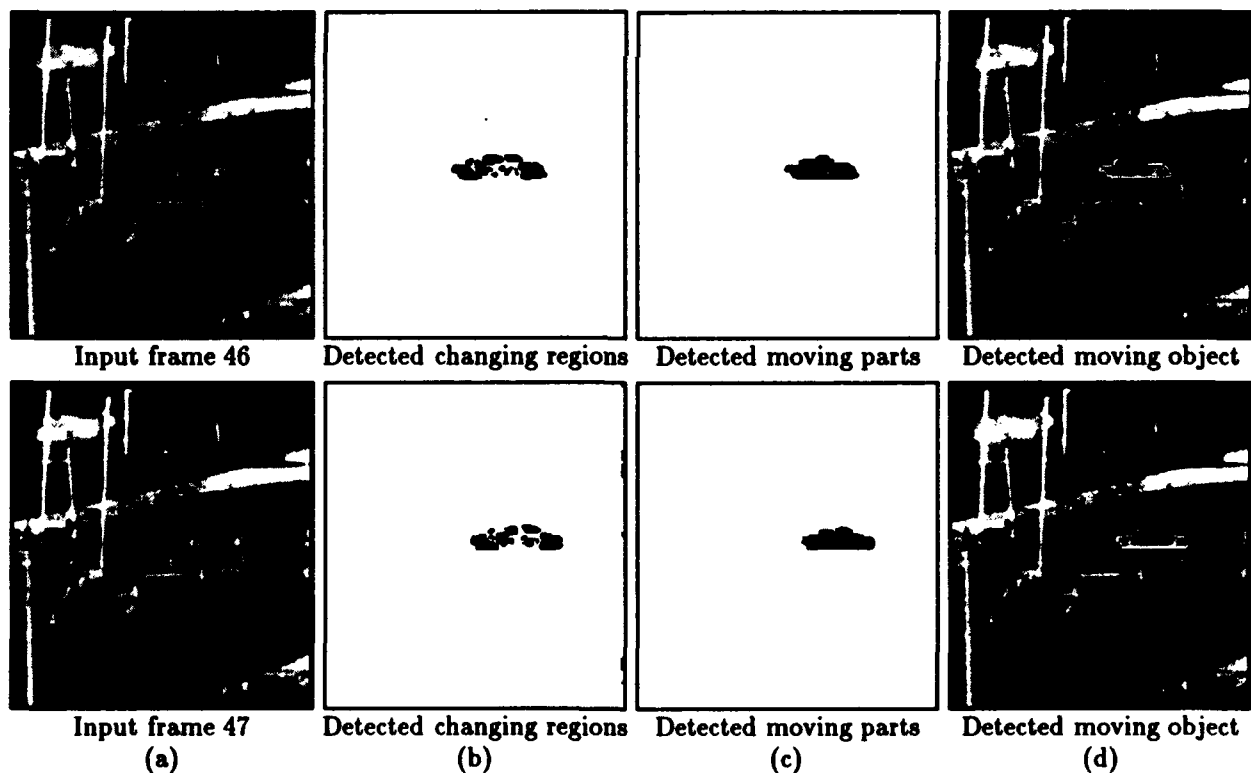


Figure 7: Moving object detection from an infra-red image sequence. (a) Input frames. (b) Changing regions segmented from camera motion compensated frame differences. (c) Moving part detected using our algorithm. (d) Superimposition of the contour of the detected moving part on the image to show the correctness of motion detection.

ages. The images in this sequence exhibit perspective deformations of close objects. Figure 5.a shows the feature points detected automatically in the first frame⁴ Figures 5.b and 5.c show the trajectories up to frames 6 and 10 respectively.⁵ The tracking is successful.

5.2 Motion Detection

Figures 6 and 7 show two motion detection results obtained by our algorithm. Figure 6 is the detection of a flying helicopter from an image sequence taken from another helicopter. Figure 7 is the detection of a moving automobile from an infra-red image sequence. In both examples, (a) shows two frames of the input image sequence; (b) shows the changing parts segmented from the camera motion compensated frame difference; (c) shows the detected object. The estimated velocity for the moving helicopter is $(u, v) = (-2.94, 0.02)$ from frames #6 to #7 and $(u, v) = (-2.92, 1.26)$ from frames #7 to #8. The estimated velocity for the moving car is $(u, v) = (15.56, 0.47)$ from frames #45 to #46 and $(u, v) = (13.87, 0.36)$ from frames #46 to #47. (d) shows the contours of detected moving objects superimposed on the camera motion compensated images. As we see, in spite of the complex background, poor image quality, and unknown camera motion, motion detection is quite good. The moving car is detected correctly from the infra-red image sequence. In the flying helicopter sequence, some parts of the background are also detected as belonging to the moving object. This is due to the fact that these background areas are uniform, yielding zeros in the translated frame difference.

Acknowledgment

The authors are grateful to Professor B.S. Manjunath for letting us use his feature extraction program, and to the computer vision research group at the University of Massachusetts for providing the Rocket Sequence. The NASA Sequence was made available through the courtesy of NASA Ames Research Center.

References

- [1] T.S. Huang, *Image Sequence Analysis*, Springer-Verlag, Berlin, 1981.
- [2] R. Kumar and A. Hanson, "Pose refinement: Application to model extension and sensitivity to camera parameters," in *Proc. DARPA Image Understanding Workshop* (Pittsburgh, PA), pp. 660-669, 1990.
- [3] H.S. Lim and T.O. Binford, "Structural correspondence in stereo vision," in *Proc. DARPA Image Understanding Workshop* (Cambridge, MA), pp. 794-808, 1988.
- [4] B.S. Manjunath, R. Chellappa, and C. Malsburg, "A feature based approach to face recognition," in *Proc. IEEE Conf. Computer Vision Pattern Recognition* (Champaign, IL), pp. 373-378, 1992.
- [5] L. Matthies, R. Szeliski, and T. Kanade, "Kalman filter-based algorithms for estimating depth from image sequences," in *Proc. DARPA Image Understanding Workshop* (Cambridge, MA), pp. 199-213, 1988.
- [6] S. Ranade and A. Rosenfeld, "Point pattern matching by relaxation," *Pattern Recognition*, Vol. 12, pp. 269-275, 1980.
- [7] I.K. Sethi and R. Jain, "Finding trajectories of feature points in a monocular image sequence," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-9, pp. 56-73, 1987.
- [8] C. Shekhar and R. Chellappa, "Visual motion analysis for autonomous navigation," Tech. Rep. CAR-TR-607/CS-TR-2862, Center for Automation Research, University of Maryland, College Park, MD 20742, March 1992.
- [9] P. Smith, B. Sridhar, and B. Hussein, "Vision-based range estimation using helicopter flight data," Tech. Rep. NASA-TM, NASA Ames Research Center, Moffett Field, CA, 1991.
- [10] Q. Tian and M.N. Huhns, "Algorithms for subpixel registration," *Computer Vision, Graphics, Image Processing*, Vol. 35, pp. 220-233, 1986.
- [11] T. Wu and R. Chellappa, "Experiments on estimating motion and structure parameters using long monocular image sequences," in *Proc. Intl. Conf. Computer Vision*, 1993.
- [12] Q. Zheng and R. Chellappa, "Estimation of illuminant direction, albedo and shape from shading," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-13, pp. 680-702, 1991.
- [13] Q. Zheng and R. Chellappa, "Automatic feature point extraction and tracking in images sequences for arbitrary camera motion," Tech. Rep. CAR-TR-628, Center for Automation Research, University of Maryland, College Park, MD 20742, June 1992.
- [14] Q. Zheng and R. Chellappa, "A computational vision approach to image registration," in *Proc. Intl. Conf. Pattern Recognition* (The Hague, The Netherlands), Aug. 1992.

⁴Originally many points are detected around the labeling letters at the top left corners of the images. These points are removed from our feature list by requiring the locations of valid feature points to be below a certain line in the image domain. Among the removed feature points, there are points on the body of the helicopter.

⁵The original images are 512×512 with black bars on the bottom of each frame. In Figure 5 only the top 512×412 portions are displayed.

Optical Flow from 1D Correlation: Application to a simple Time-To-Crash Detector

†Nicola Ancona and †Tomaso Poggio

†Tecnopolis CSATA Novus Ortus - Robotic & Automation Laboratory
70010 Valenzano, Bari - Italy

†Massachusetts Institute of Technology - Artificial Intelligence Laboratory
545 Technology Square, Cambridge, MA 02139 - U.S.A.

Abstract

In the first part of this paper we show that a new technique exploiting 1D correlation of 2D or even 1D patches between successive frames may be sufficient to compute a satisfactory estimation of the optical flow field. The algorithm is well-suited to VLSI implementations. The sparse measurements provided by the technique can be used to compute qualitative properties of the flow for a number of different visual tasks. In particular, the second part of the paper shows how to combine our 1D correlation technique with a scheme for detecting expansion or rotation ([1]) in a simple algorithm which also suggests interesting biological implications. The algorithm provides a rough estimate of time-to-crash. It was tested on real image sequences. We show its performance and compare the results to previous approaches.

1 Introduction

The problem of how to compute efficiently estimates of the optical flow at sparse locations is of critical importance for practical implementations in a number of different tasks. A specific example is the detection of expansion of the visual field with a rough estimate of *time-to-crash* (TTC). The question has also interesting relations with biology, as we will discuss later. In this paper we propose an efficient algorithm for computing the optical flow which performs well in a number of experiments with sequences of real images and is well suited to a VLSI implementation.

Optical flow algorithms based on patchwise correlation of filtered images perform in a satisfactory way [2] and better in practice than most other approaches (see [3]). Their main drawback is computational complexity that forbid at present useful VLSI implementations. In this paper we show that 1D patchwise correlation may provide a sufficiently accurate estimate of the optical flow¹.

¹In this paper we use mainly the L_2 distance rather than the correlation itself. Since the two measures are equivalent

We will then show with experiments on real image sequences how to apply this technique to measure time-to-crash, by exploiting a recently proposed scheme [1]. The latter scheme, which is robust and invariant to the position of the focus of expansion or the center of rotation, relies on sparse measurements of either the normal or the tangential component of the optical flow (relative to a closed contour). We will also discuss some broad implications of this work for the practical computation of the optical flow and for biology, in particular its relation to Reichardt's-type models.

There are two main and quite separate contributions in this paper:

1. an efficient 1D correlation scheme to estimate the optical flow along a desired direction
2. the experimental demonstration that a previously proposed algorithm for estimating time-to-crash performs satisfactorily in a series of experiments with real images in which the elementary measurements of the flow are obtained by the new 1D correlation scheme.

2 Computing the Optical Flow along a Direction

How can the component of the optical flow be measured efficiently along a certain desired direction? As argued by Verri and Poggio [4] a qualitative estimate is often sufficient for many visual tasks. For the task of detecting a potential crash, for instance, it has been suggested ([1]) that a *precise* measurement of the normal component of the flow may not be necessary, since the precise definition of the optical flow is itself somewhat arbitrary: it is sufficient that the estimate be qualitatively consistent with the values of the perspective 2D projection of the "true" 3D velocity field *for the particular stimulus*. In other words, even estimates that don't really measure image-plane velocity (like Reichardt's correlation model or equivalent energy models), since they also

for the purposes of this paper, we will often use the terms "correlation" and "distance" in an interchangeable way.

depend on spatial structure of the image, may be acceptable for several visual tasks, if their estimates are consistent over the visual field. Certain uses of a crash detector are good examples. It turns out that even a rough estimate of time-to-crash (TTC) is possible using approximate estimates of the optical flow field. Flies and other insects rely for landing on what appears to be a qualitative estimate of the time-to-crash!

2.1 1D correlation of 2D patches

A possible approach for an approximative estimate of the optical flow is to use a *1D correlation scheme* between two successive frames, instead of 2D correlation, as in [2]. The basic idea underlying the full 2D correlation technique that we label *2D-2D* in this paper² is to measure, for each desired location, the (x, y) shift that maximizes the correlation between 2D patches centered around the desired location in successive frames. The patchwise correlation between the image at time t and at time $t + \delta t$ is defined as

$$\Phi(\delta x, \delta y; t) \equiv I^w \otimes I = \int I^w(\xi, \eta; t) I(\delta x + \xi, \delta y + \eta; t + \delta t) d\xi d\eta \quad (1)$$

where $I^w(\xi, \eta; t)$ is the image at time t windowed to the patch of interest and set to 0 outside it. The L_2 distance has very similar properties to the correlation measure³. In the context of this paper, minimizing the L_2 distance is exactly equivalent to maximizing the correlation (the observation is due to F. Girosi). As noticed before [2], the previous idea can be regarded as an approximation of a regularization solution to the problem of computing the optical flow⁴. Usually, one does not use grey values directly but rather some filtered version of the image, for instance through a Laplacian-of-a-Gaussian filter (see [2]), possibly at different resolutions.

Let us call $D(\delta x, \delta y)$ the L_2 distance between 2 patches in 2 frames at location (x, y) as a function of the shift vector $(\delta x, \delta y)$. The "winner-take-all" scheme finds $s^* = (\delta x^*, \delta y^*)$ that minimizes D (or maximize the correlation function $\Phi(\delta x, \delta y)$) and assumes that the optical flow estimate is $u^* = s^* / \Delta t$, where Δt is the interframe interval.

It is natural to consider whether the component of u^* along a given direction, for instance x , may be estimated in a satisfactory way simply by computing the δx that minimizes $D(\delta x, 0)$, that is the patchwise correlation as a function of x shifts only. We have found in our experiments that 1D correlation of a 2D patch provides estimates of δx^* that are very close to the estimates obtained from the *2D-2D* technique. We label this technique *1D-2D*, since it involves one-dimensional correlations on 2D patches.

²It is also called winner-take-all method.

³The L_2 distance is in this case the square root of the sum of the squares of the differences between values of corresponding pixels. Other "robust" distance metric may be used, such as the sum of absolute values.

⁴And in turn several definitions of the optical flow such as Horn and Schunk's, can be shown to be approximations of the correlation technique [5].

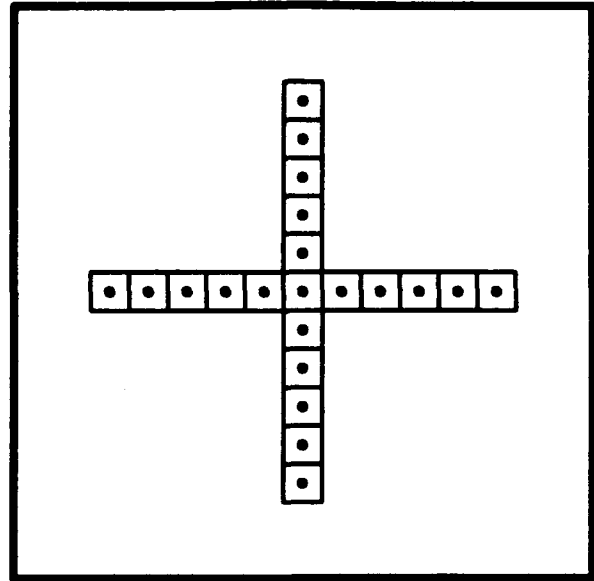


Figure 1: The search space for the *1D-2D* scheme used for the computation of the x and y components of the optical flow.

If we combine horizontal and vertical motion detectors of our 1D, winner-take-all type (see fig.1), we obtain an appealing scheme to estimate the optical flow field at one point. The optical flow in one point is the vector sum of the x and y components computed by using such motion detectors. The key aspect of this approach is its reduction of the complexity of the problem, while maintaining a good estimation of the flow field: a complete two-dimensional search required in the winner-take-all scheme [2] is reduced to two one-dimensional searches. Let us call v_{max} the maximum velocity expected on the image plane. In [2] the search space size to scan is $(2v_{max} + 1)^2$ for each point; in our approach, its size is limited to $2(2v_{max} + 1)$.

2.2 1D correlation of 1D patches

So far we have discussed that 1D correlation of 2D patches gives a satisfactory estimate of the optical flow between two successive frames, reducing the search space of corresponding points. This is equivalent to saying that the

$$\min_{\delta x} \Phi(\delta x, 0)$$

and

$$\min_{\delta y} \Phi(0, \delta y)$$

give a satisfactory estimate of

$$\min_{\delta x, \delta y} \Phi(\delta x, \delta y).$$

This suggests a further simplification: instead of $\Phi(\delta x, 0)$ consider a projection on x of $\Phi(\delta x, \delta y)$ obtained by some form of averaging operation on y , that is

$$\Phi * h_2$$

where h_2 is a 2D filter such as a Gaussian elongated in the y direction and $*$ stands for the convolution operator. By well known properties of the Gaussian function, h_2 can always be written as

$$h_2 = h * h,$$

where h are Gaussian functions of appropriate variance. Assuming that we can neglect the patch size in the definition of Φ , we can write:

$$\Phi * h_2 = (I_t * h) \otimes (I_{t+t} * h) \quad (2)$$

where $I_t = I(x, y, t)$.

Thus, in the approximation of a large patch size, projecting the correlation function is equivalent to appropriately filtering the two images before correlation. Since it is usually better to discount the average intensity as well as small gradients through a high-pass filtering operation, in order to estimate the x -component of u , we just perform a Gaussian smoothing in the y direction, as shown in eq. 2, and then perform an additional convolution with the first or second derivative of a Gaussian function elongated in the x direction. Therefore the intensity function that is used in practice in the correlation operation is:

$$\hat{I}_t = (G_{\sigma_y}(y) * I_t) * G''_{\sigma_x}(x) \quad (3)$$

where σ_x and σ_y define the receptive field of such an elementary motion detector. After this filtering step, it is sufficient to evaluate the maximum of the correlation function only on 1D patches to obtain an estimate of the x component of the flow. The previous argument does not strictly apply to the L_2 distance measure that we have used in our experiments. The very close similarity between correlation and distance, however, suggests a very similar behavior in both cases. We label this technique the *1D-1D* scheme since it involves 1D correlations of 1D patches.

3 A crash detector: the Green theorem scheme

As described in [1] (see also [6]), the divergence of the optical field $\nabla u(x, y)$ is a differential measure of the local expansion ($\nabla u(x, y) = \frac{\partial u_x(x, y)}{\partial x} + \frac{\partial u_y(x, y)}{\partial y}$). For a linear field (i.e. $u(x) = Ax$), the divergence of u is the same everywhere. In the case of linear fields (and all fields can be approximated by linear fields close to the singularity), the integral of the divergence over an area is invariant with respect to the position of the center of expansion. Green's theorems show that the integral over a surface patch S of the divergence of a field u is equal to the integral along the patch boundary of the component of the field which is normal to the boundary ($u \cdot n$). In formula

$$\int_S \nabla u(x, y) dx dy = \int_C u \cdot n dl. \quad (4)$$

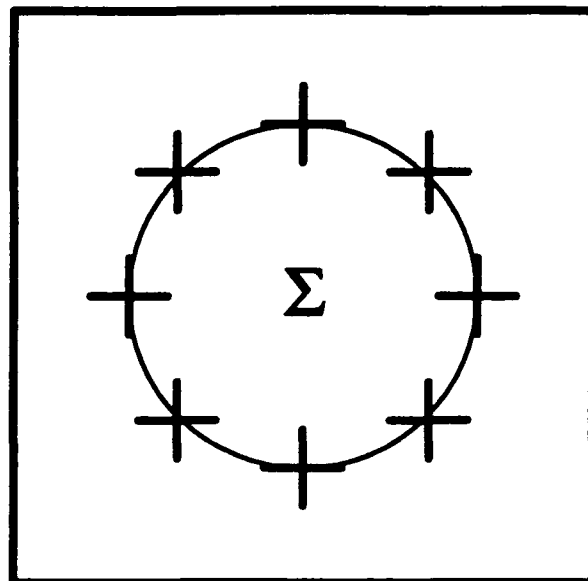


Figure 2: A TTC detector consisting of elementary motion detectors (see figure 1) at several locations along a closed contour. Each of the elementary motion detectors could be replaced by a single detector normal to the circle.

Therefore, since for a linear field $\nabla u = 2/\tau$ where τ is the time to crash (TTC), a TTC detector that exploits the Green theorem just needs to sum over a closed contour, say a circle, the normal component of the flow measured at n points along the contour. We assume that the task is to compute *time to crash* (TTC) for pure translational motion. Possibly the simplest TTC detector of this type, shown in figure 3, is composed of just 4 elementary motion detectors. In this case we have to sum the x -component of u for the horizontal detectors and the y -component of u for the vertical ones, with the correct sign.

Due to the invariance with respect to the position of the focus of expansion (or contraction) we can in principle arrange a certain number of them (see fig.4) on the image plane. Our simulations suggest that one detector with a large radius (fig. 3) is better than several, "smaller" detectors (fig. 4) in situations in which the whole visual field expands, probably because of better numerical stability of the estimates. Of course, a "large" detector has a poorer spatial resolution and this may be a problem in some applications (but not ours).

We have discussed so far schemes for detecting expansion. Similar arguments hold for rotation. The Green's theorem relevant to this case is usually called Stokes' theorem and takes the form

$$\int_S \nabla \wedge u(x, y) \cdot dS = \int_C u \cdot dr \quad (5)$$

which says that the total flux of the differential measure of "rotationality" of the field $\nabla \wedge u$ across the surface patch S is equal to the integral along the boundary C of the surface patch of the component of the field which is tangential to the boundary. As described in [1], each

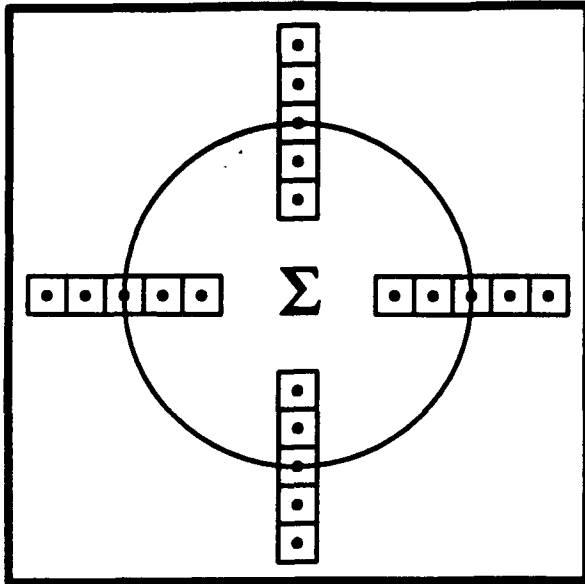


Figure 3: Time-to-crash detector that exploits Green theorem.

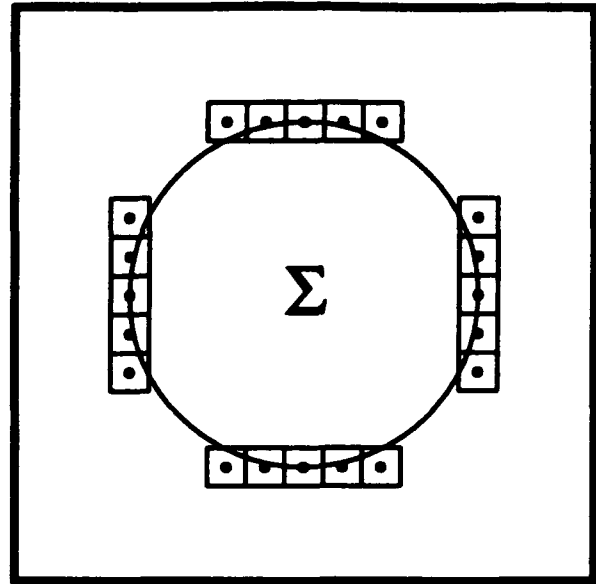


Figure 5: Motion detector that exploits Stokes' theorem.

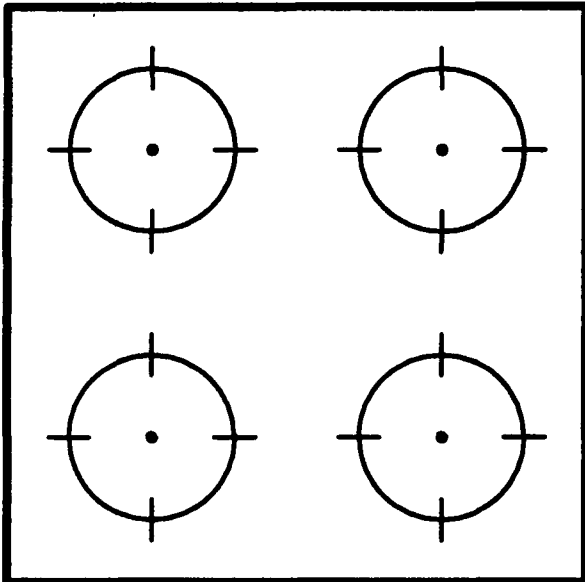


Figure 4: A possible arrangement of TTC detectors in the image plane that is not as efficient as a single TTC detector with greater radius but has higher spatial resolution.

elementary detector evaluates the tangential flow component at the contour of the receptive field (see fig.5). In this case a detector has to compute the component of u along the tangential direction at the contour.

4 Experimental results

4.1 The 1D-2D scheme

We have extensively tested our approach on real image sequences. Each sequence was acquired from a camera mounted on mobile platform moving at constant velocity.

In all experiments the movement of the vehicle was a forward translation along a straight trajectory. We have verified the results obtained from our 1D-2D approach with the standard winner-take-all (2D-2D) scheme [2] [3].

Figure 9 shows the first and last image of a sequence composed of 100 frames. Each image of the sequence is first convolved with a Gaussian filter having $\sigma = 0.5$. In both the algorithms we have used $v_{max} = 9$ and $\nu = 20$ pixels, where v_{max} is the maximum expected velocity of the points on the image plane and ν is the ray of the patch used for the evaluation of Φ . In other words, the correlation window used for the optical flow computation is 41×41 pixels and the search space used is 19×19 by 2D-2D and $19 + 19$ by 1D-2D. Figures 10 shows the optical flows computed by the two methods using two successive images of the sequence. The position of the focus of expansion was computed by using the approach described in [7].

We have used the method described in [7] and [1] to verify the TTC estimation. To compute the TTC at a point by using the method in [7], we used an area of 81×81 pixels around that point. The points were 10 pixels apart. To compute TTC by using the method described in [1], we used a lattice of overlapping motion detectors. The distance between two points on the lattice was 10 pixels. Each detectors had a receptive field of ray $r = 40$ pixels. In fig. 11, we compare the results obtained by using the 2D-2D estimation of the optical flow with the 1D-2D one, by using the two different methods in the first stage of the TTC. Performing a linear best fit on the TTC measurements, we obtain a slope of $m = -1.036$ by using the optical flows computed by 2D-2D and the method described in [7], and $m = -1.139$ by using the optical flows computed by 1D-2D and the method described in [1]. Comparing the true TTC (straight line in fig. 11) with the TTC measures obtained by using the second method, we estimate an absolute error in the

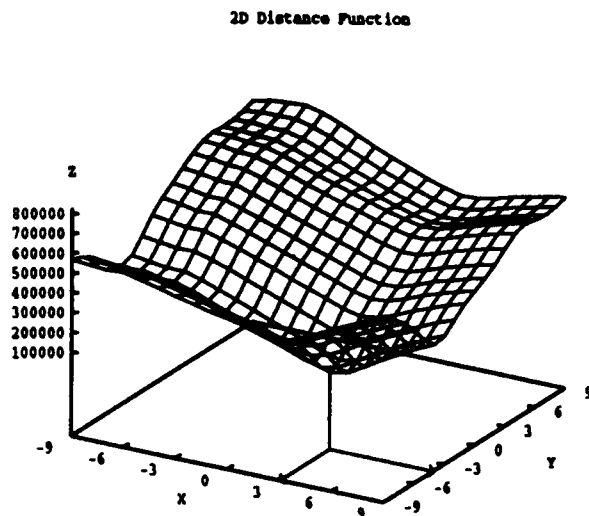


Figure 6: 2D distance function. The arrows indicate the position of the minimum.

mean of 2.63, with a standard deviation of 3.35 frame unit. In terms of relative units, the error in the mean is 5.7% with a standard deviation of 6.1%.

4.2 The 1D-1D scheme

In this section we compare the results obtained by using 2D-2D and 1D-1D. In both techniques, we have used $v_{max} = 9$ and $\nu = 20$ pixels. In other words, the correlation window used for the optical flow estimation is 41×41 pixels for 2D-2D and 41 pixels for 1D-1D. In the filtering step we have used $\sigma_y = 6$ and $\sigma_x = 3$ pixels for computing the x -component of the optical flow. These values of σ produce a receptive field of an elementary motion detector equals to that used by 2D-2D (1681 pixels). The fig. (6) shows a plot of the 2D correlation function used in 2D-2D over a 2D search space and a 2D integration area. Figures (7) and (8) show a plot of 1D correlation functions used by the 1D-1D technique to estimate both components of the optical flow. In this case we used two 1D search spaces (in x and y directions respectively) and a 1D integration area. Notice that this approach is capable of computing a reliable estimation of the flow vectors, while reducing the complexity of the problem.

Figures (13), (17), (21) show the first and the last image of three sequences acquired from a camera mounted on a mobile platform moving at constant velocity, along a straight trajectory. Figures (14), (18), (22) show the optical flows computed by the two methods, by using two successive frames of the sequences. The mean (continuous line) and the standard deviation (dashed line) of the error on the optical flow estimation is shown in figures (15), (19), (23). Figures (16), (20), (24) show the TTC estimation by using the two different methods. In each experiments, we have used only one TTC detec-

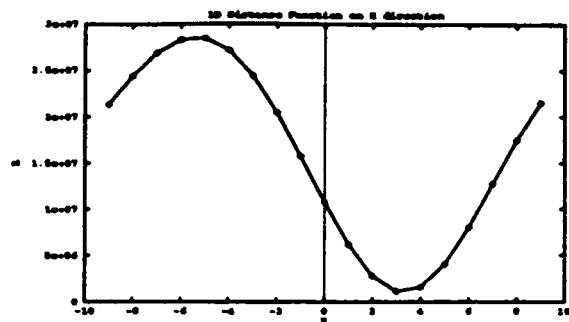


Figure 7: 1D distance function computed on the x direction by using 1D-1D.

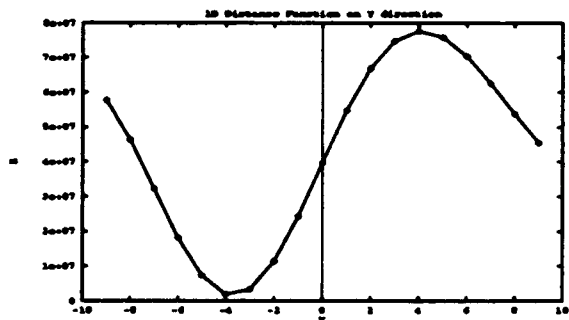


Figure 8: 1D distance function computed on the y direction by using 1D-1D.

tor, with receptive field of $r = 80$ pixel, composed by 32 elementary motion detectors (see fig. (2)).

5 Conclusions

5.1 Extensions of the optical flow algorithm

There are several directions in which we plan to improve and extend our scheme:

- it may be possible to reduce further the number of sample points for D_p (i.e. the number of shifts) by using techniques for learning from examples such as the RBF technique ([8]) to approximate $D_p(\delta x)$ as $D_p(\delta x) = \sum c_n G(x - t_n)$, and then find the minimum of D_p in terms of the dynamical system $dx/dt = -\epsilon \sum c_n G'(x - t_n)$. An alternative strategy is to try to learn directly the function $\min D_p(x)$ from the samples of D_p , using a few examples of D_p "typical" for the specific situation. The conjecture is that the RBF technique may be able to learn the mapping $\min D_p(x)$ from examples of functions of the same class (compare Poggio and Vetter, 1992). A similar idea is to try to learn how to sample the correlation function as a function of past sampled values. Again, the training examples would be functions of the same class. This would provide at each t an estimate of the most appropriate correlation shifts to try.
- instead of simply measuring $D_{i,i-1}$, that is the distance between frame i and frame $i-1$, we could mea-

sure in addition also $D_{i,i-2}$, $D_{i,i-3}$, ...and combine them in an estimate of the optical flow component by taking the average of $D_{i,i-1}/\Delta t$, $D_{i,i-2}/2\Delta t$, $D_{i,i-3}/3\Delta t$, etc. This technique may be improved further by using a Kalman filter.

- the same basic scheme of figure 1 may be used to compute horizontal and vertical disparities among the two frames of a stereo pair.
- confidence measures will be developed to further improve the performance of the technique.

5.2 Biological implications of our 1D technique

Poggio et al. ([1]) conjectured that "the specific type of elementary motion detectors that are used to provide the estimate of the normal component of the flow is probably not critical. Radially oriented (for expansion and contraction), two input elementary motion detectors such as the correlation model [9; 10; 11; 12] – or approximations of it are likely to be adequate. The critical property is that they should measure motion with the correct sign." Our results confirming their conjecture (since they suggest that 1D correlation (or L_2 distance estimation) are sufficient for an adequate estimate of qualitative properties of the optical flow) have interesting implications for biology. Consider a 2D array of Reichardt's detectors (for motion in the x direction) with spacing Δx and also detectors with spacings $2\Delta x$ etc. Take the sum of all detectors with the same spacing over a 2D patch. Perform a winner-take-all operation on these sums. Select the set with optimal spacing as the one corresponding to the present estimate of optical flow. This scheme is analog in time but otherwise equivalent to the one we have implemented. In formulae

$$\sum (I_i(t) - I_{i+k}(t - \Delta t))^2$$

where Δt is the interframe interval in our implementation and is the delay in Reichardt's model⁵, k represents the shift in our computation of D and represents the separation between the inputs to Reichardt's modules, $I_i(t)$ is the image value (in general spatially and temporally filtered) at location i and time t and the sum \sum is taken over the 2D patch of detectors of the same type.

Thus an array of Reichardt's models with different spacings of the 2 inputs (in x) could be used in a plausible way to estimate the optical flow component along the direction of the two-inputs detectors. Notice that a plausible implementation in terms of Reichardt's detectors of the 2D correlation based algorithm would be much harder, since it would effectively require detectors with all possible 2D spacings. This seems implausible and contrary to experimental evidence in the fly, where only a small number of separations and directions (as small as 3) seem present.

The above description is equivalent to our 1D-2D scheme and involves the summation over x and y "patches" of elementary 1D motion detectors. In the

⁵We have written here the quadratic version of Reichardt's model; the same argument carries over to the standard model with multiplication: for the basic equivalence of the the quadratic and multiplication version see [11])

fly this is plausible, given the known summation properties of specific wide field lobula plate cells⁶. Our 1D-1D scheme on the other hand would require a summation over the x dimension only (in our example) but an oriented filtering of the image with receptive fields elongated in y before the elementary motion detectors. It is possible that this second scheme may be used in the fly by different summation cells with smaller receptive fields. It is also possible that the wide field lobula plate cells effectively implement a scheme between the 1D-2D and the 1D-1D by using some oriented filtering before motion detection and limited y integration of the output of the elementary motion detectors. Similar considerations may apply to some of the motion selective cortical cells.

5.3 The Time-to-Crash detector

The TTC detector we have simulated is not the only possible scheme. Others are possible (see for instance [6]) that take into account more complex motions than just frontal approach to a horizontal surface.

It is also conceivable that the scheme we suggest may be simplified even further in certain situations. For instance, it may be sufficient in the summation stage to use the value of the correlation for a fixed (and reasonable) shift – instead of an estimate of the optical flow, that is the shift that maximize correlation. This is equivalent to use directly the output of Reichardt's correlation nets instead of using the result of a winner-take-all operation on a set of Reichardt's nets with different spacings (or delays).

Another related idea is to continuously adjust the correlation shifts in order to track as closely as possible the maximum of the correlation (or the minimum of the distance): in this way it may be possible to reduce the computation of the correlation to just a few shifts, especially if time-filtering techniques are also used.

Acknowledgments: We are grateful to John Harris and Federico Girosi for many discussions and very useful comments.

References

- [1] T. Poggio, A. Verri, and V. Torre. Green theorems and qualitative properties of the optical flow. Technical Report A.I. Memo No. 1289, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 1991.
- [2] J. Little, H. Bulthoff, and T. Poggio. Parallel optical flow using local voting. In *IEEE 2nd International Conference in Computer Vision*, 1988.
- [3] Koch Christof, Andrew Moore, Wyeth Bair, Timothy Horiuchi, Brooks Bishofberger, and John Lazaro. Computing motion using analog vlsi vision chips: An experimental comparison among four approaches. In *IEEE Workshop on Visual Motion*, 1991.

⁶The patch would be very large and would correspond to the receptive field of the cell, that is its integration domain

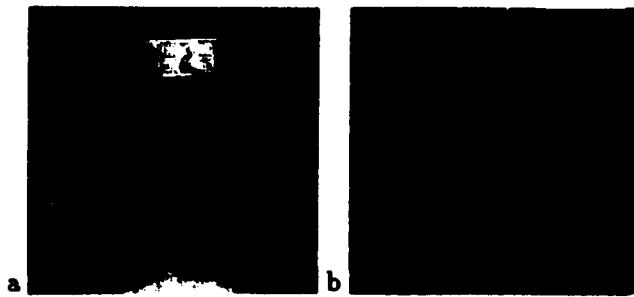


Figure 9: (a) First and (b) last image of the sequence.

- [4] A. Verri and T. Poggio. Motion field and optical flow: Qualitative properties. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:490-498, 1989.
- [5] T. Poggio, W. Yang, and V. Torre. Optical flow: Computational properties and networks, biological and analog. In *The Neuron as a Computational Unit Proceedings*, Cambridge, UK, June 1988. Scientific Applications International Corporation.
- [6] R. Cipolla and A. Blake. Surface orientation and time to contact from image divergence and deformation. In *Second European Conference on Computer Vision*, pages 187 - 202, Santa Margherita Ligure - Italy, 1992.
- [7] A. Verri and M. Campani. An algebraic procedure for the computation of optical flow from first order derivatives of the image brightness. In *IEEE Work. on Robust Comp. Vision*, Seattle, 1990.
- [8] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9), September 1990b.
- [9] W. Reichardt. Autocorrelation, a principle for evaluation of sensory information by the central nervous system. In W.A. Rosenblith, editor, *Principles of Sensory Communication*, pages 303-317, New York, 1961. John Wiley and Sons.
- [10] W. Reichardt. Evaluation of optical motion information by movement detectors. *J. Computational Physics*, 161:533-547, 1987.
- [11] Tomaso Poggio and Werner Reichardt. Considerations on models of movement detection. *Kybernetik*, 13:223-227, 1973.
- [12] J.P.H van Santen and G. Sperling. Temporal covariance model of human motion perception. *Journal of Optical Society of America - A*, 1:451-473, 1984.

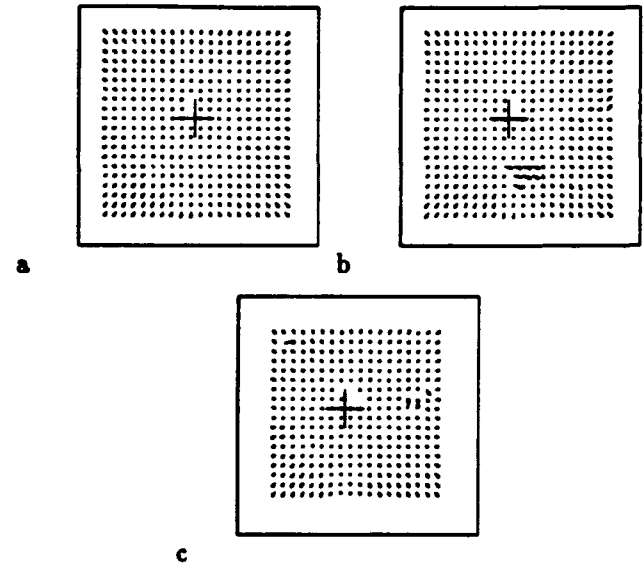


Figure 10: An example of optical flows computed by using (a) 2D-2D, (b) 1D-2D and (c) 1D-1D. In most frame pairs in a sequence the three flows are much more similar to each other.

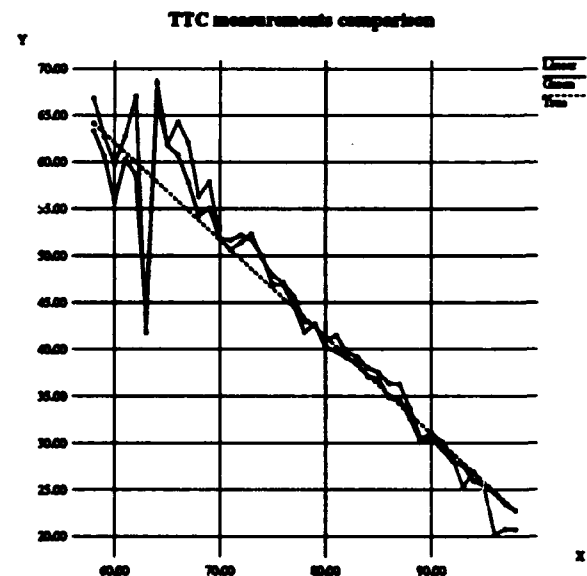


Figure 11: TTC measurements comparison by using 2D-2D and 1D-2D. In this and the following figures the abscissa gives the time in terms of elapsed frames; the ordinate gives the estimate of the time to crash in frame units.

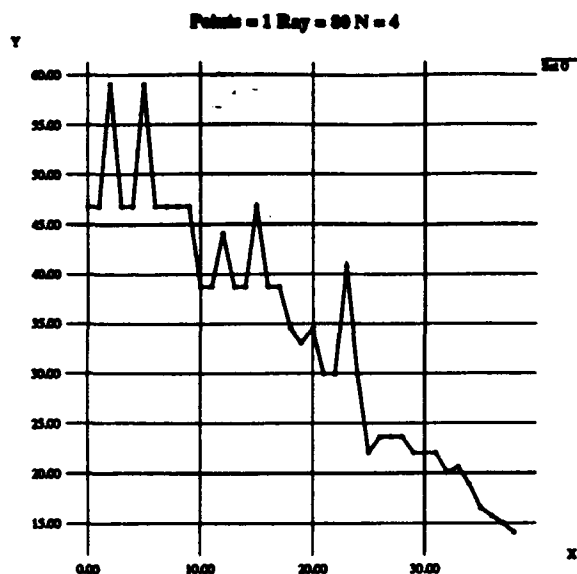


Figure 12: TTC measurements by using *1D-1D* and a TTC detector with 4 elementary motion detectors.

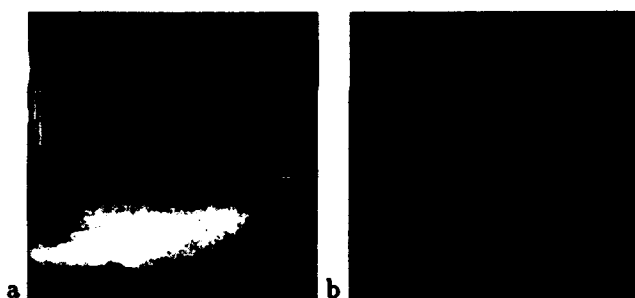


Figure 13: (a) First and (b) last image of the sequence.

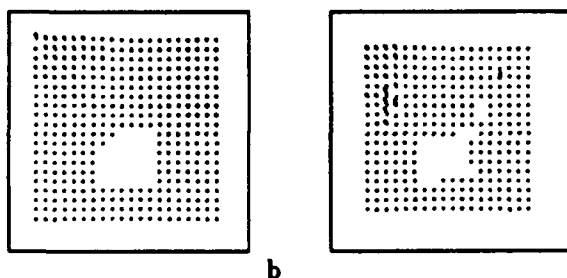


Figure 14: An example of a flow field computed at one point in time in the above sequence, obtained by using (a) *2D-2D* and (b) *1D-1D*.

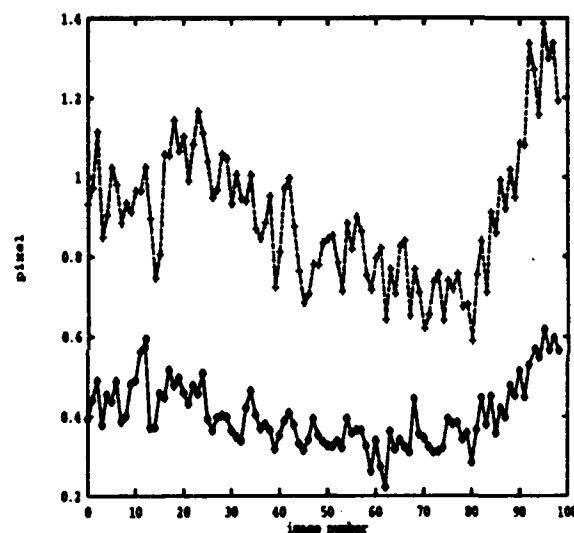


Figure 15: Mean (dotted line) and standard deviation (dashed line) of the error of the optical flow estimation.

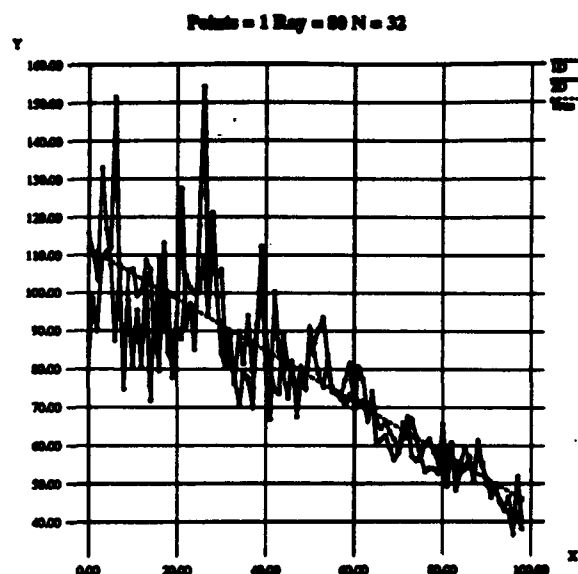


Figure 16: TTC estimation by using one TTC detector, with receptive field of $r = 80$ and 32 elementary motion detectors. The slope of the true TTC, computed by using the optical flows obtained by 2D-2D, is $m = -0.672$. The slope of the straight line, computed by using the TTC measures obtained by 1D-1D, is $m = -0.64$. A comparison of the TTC measures obtained by 1D-1D with the true TTC yields a mean absolute error of 9.02, with a standard deviation of 9.54. The relative error in the mean is 10.79% with a standard deviation of 9.49%. In order to evaluate the error in the time to crash estimation, the following steps have been performed. The true time to crash was estimated from a linear best fit of the TTC measures obtained by using the 2D-2D scheme for the optical flow estimation. The figures show the straight line that represents the theoretical behavior of the TTC. A linear best fit of the TTC measures obtained by using the 1D-1D scheme for the optical flow estimation was then performed in order to evaluate the slopes of the two straight lines. The absolute and relative error between the "true" TTC and the one measured by the 1D-1D scheme was then estimated. Let us call τ^* the true TTC. The absolute error is $E_a = |\tau^* - \tau|$ and the relative error is $E_r = |\tau^* - \tau|/|\tau|$.

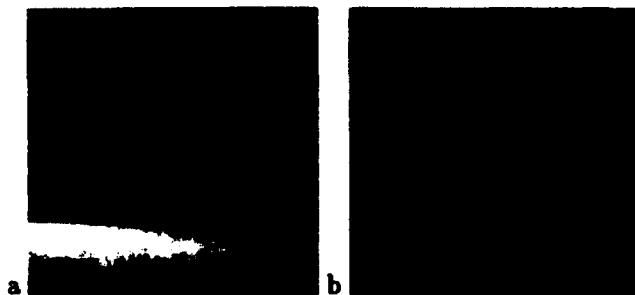


Figure 17: (a) First and (b) last image of the sequence.

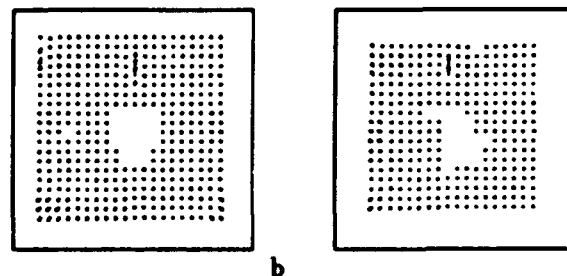


Figure 18: An example of a flow field obtained by using (a) 2D-2D and (b) 1D-1D.

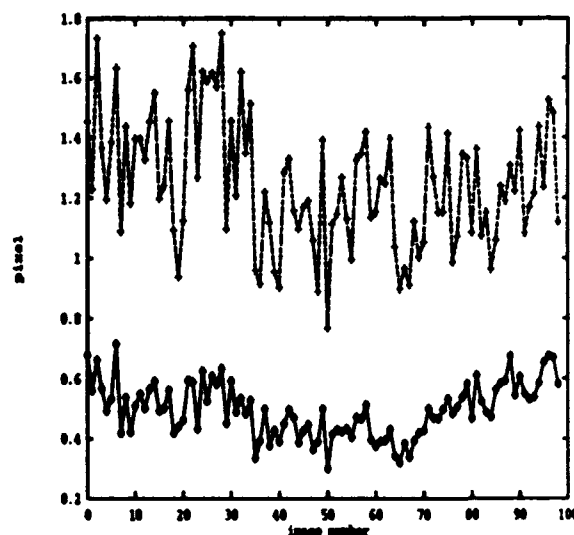


Figure 19: Mean (dotted line) and standard deviation (dashed line) of the error relative to optical flow estimation.

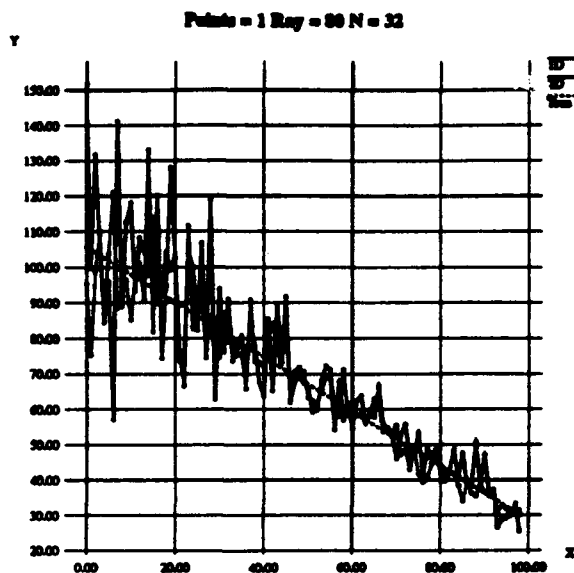


Figure 20: TTC estimation by using one TTC detector, with receptive field of $r = 80$ and 32 elementary motion detectors. The slope of the *true* TTC, computed by using the optical flows obtained by *2D-2D*, is $m = -0.77$. The slope of the straight line, computed by using the TTC measures obtained by *1D-1D*, is $m = -0.83$. Comparing the TTC measures obtained by *1D-1D* with the true TTC, we had a mean absolute error of 8.02, with a standard deviation of 8.97. With respect to the relative error we had a mean of 10.9% and a standard deviation of 9.72%.

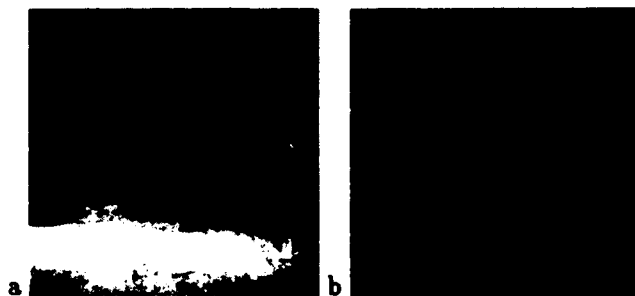


Figure 21: (a) First and (b) last image of the sequence.

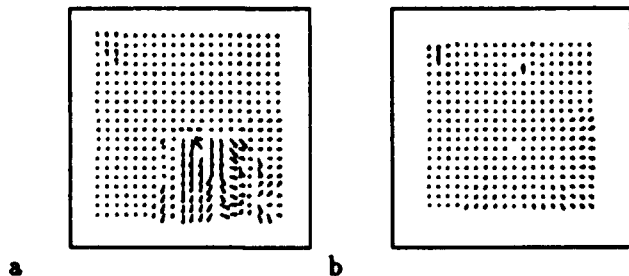


Figure 22: Flow field obtained by using (a) *2D-2D* and (b) *1D-1D*.

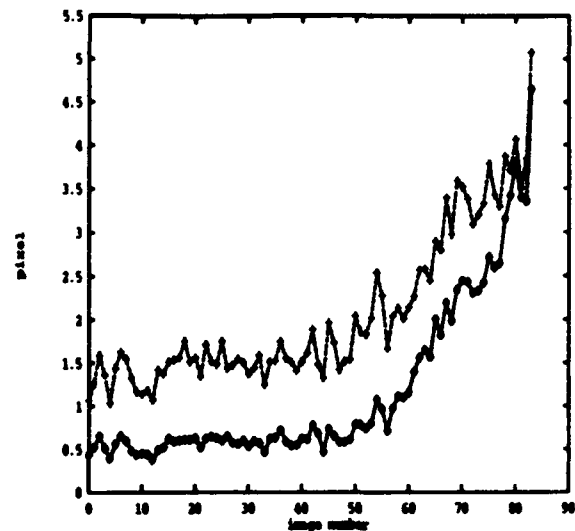


Figure 23: Mean (dotted line) and standard deviation (dashed line) of the error relative to optical flow estimation.

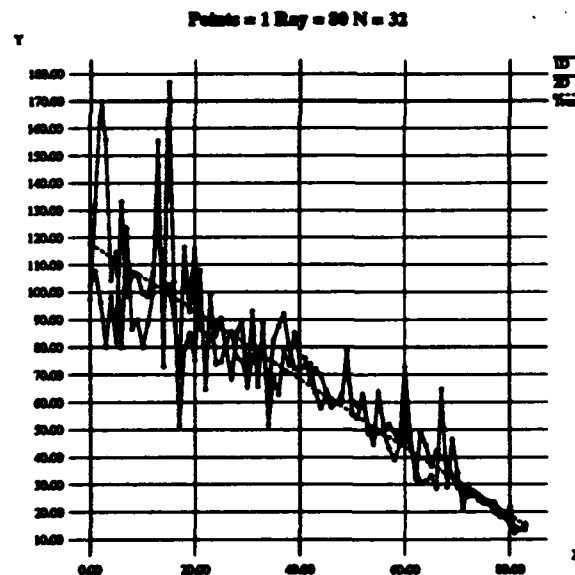


Figure 24: TTC estimation by using one TTC detector, with receptive field of $r = 80$ and 32 elementary motion detectors. The slope of the *true* TTC, computed by using the optical flows obtained by *wtg-2D*, is $m = -1.24$. The slope of the straight line, computed by using the TTC measures obtained by *1D-1D*, is $m = -1.14$. Comparing the TTC measures obtained by *1D-1D* with the true TTC, we had a mean absolute error of 7.6, with a standard deviation of 7.9. With respect to the relative error we had a mean of 11.4% and a standard deviation of 10.3%.

A Paraperspective Factorization Method for Shape and Motion Recovery

Conrad J. Poelman and Takeo Kanade

School of Computer Science

Carnegie Mellon University

5000 Forbes Avenue

Pittsburgh, PA 15213-3890

(Conrad.Poelman@cs.cmu.edu, tk@cs.cmu.edu)

Abstract

In this paper, we present a method for recovering both the shape of an object and its motion relative to the camera, from a sequence of images of the object, using feature points tracked throughout the sequence. Our method uses a projection model known as paraperspective projection, which closely approximates perspective projection by modelling two effects not modelled under orthographic projection; the apparent change in size of an object as it moves along the camera's optical axis, and the different angle from which an object is viewed as it moves parallel to the image plane. Our paraperspective factorization method can be applied to a wide range of motion scenarios, and can recover the distance from the camera to the object in each image. The method assumes no model of the motion or of the object's shape, and recovers the shape and motion accurately even for distant objects.¹

1. Introduction

Recovering the geometry of a scene and the motion of the camera from a stream of images is an important task in a variety of applications, including navigation, robotic manipulation, and aerial cartography. While this is possible in principle, traditional methods have failed to produce reliable results in many situations [Broida et al., 1990].

Tomasi and Kanade [1991a, 1991b] developed a robust and efficient method for accurately recovering the shape and motion of an object from a sequence of images using extracted feature points.

1. This research was partially supported by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C1465, ARPA Order No. 7597.

Their method uses an orthographic projection model, which is described by linear equations. It achieves its accuracy and robustness by using a large number of images and feature points, and by directly computing shape without computing the depth as an intermediate step. The method was tested on a variety of real and synthetic images, and was shown to perform well even for distant objects.

There are, however, some limitations of the method due to its use of the orthographic projection model. The model contains no notion at all of the distance from the camera to the object. As a result, image sequences containing large translations toward or away from the camera often produce deformed object shapes, as the method tries to explain the size differences in the images by creating size differences in the object. It also supplies no estimation of translation along the camera's optical axis, which limits its usefulness for certain tasks.

Fortunately, there exist several perspective approximations which capture more of the effects of perspective projection while remaining linear. Scaled orthographic projection, sometimes referred to as "weak perspective" [Mundy and Zisserman, 1992], accounts for the scaling effect of an object as it moves towards and away from the camera. Paraperspective projection, first introduced by Ohta [1981] and named by Aloimonos [1990], accounts for the scaling effect as well as the different angle from which an object is viewed as it moves in a direction parallel to the image plane.

In this paper, we present a new factorization method based on the paraperspective projection model. The paraperspective factorization method takes a set of points extracted from the images and tracked from one image to the next, and computes the shape of the object and the motion of the camera. The method is still fast and robust with respect

to noise. However, it can be applied to a wider realm of situations than the original factorization method, such as sequences containing significant depth translation or containing objects close to the camera, and can be used in applications where it is important to recover the distance to the object in each image, such as navigation.

We begin by describing our camera and world reference frames and introduce the mathematical notation that we use. We review the original factorization method as defined in [Tomasi and Kanade, 1991b], presenting it in a slightly different manner in order to make its relation to the paraperspective method more apparent. We then present our paraperspective factorization method. We conclude with the results of some experiments which demonstrate the practicality of our system.

2. Problem Description

In a shape-from-motion problem, we are given a sequence of F images taken from a camera that is moving relative to an object. We locate P prominent feature points in the first image, and track these points from each image to the next, recording the coordinates (x_p, y_p) of each point p in each image f . Each feature point p that we track corresponds to a single world point, located at position s_p in some fixed world coordinate system. Each image f was taken at some camera orientation, which we describe by the orthonormal unit vectors i_f, j_f , and k_f , where k_f points along the camera's line of sight, i_f corresponds to the camera image plane's x-axis, and j_f corresponds to the camera image's y-axis. We describe the position of the camera in each frame f by the vector t_f pointing to the camera's focal point. This formulation is illustrated in Figure 1.

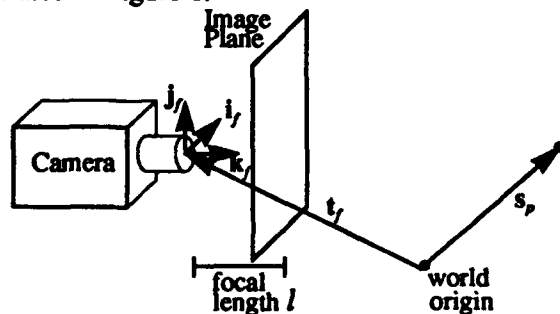


Figure 1. Coordinate system

The result of the feature tracker is a set of P feature point coordinates (x_p, y_p) for each of the F frames of the image sequence. From this information, our

goal is to recover the estimated shape of the object, given by the position s_p of every point, and the estimated motion of the camera, given by i_f, j_f, k_f , and t_f for each frame in the sequence. Rather than recover t_f in world coordinates, we generally recover the three separate components $t_f \cdot i_f, t_f \cdot j_f$, and $t_f \cdot k_f$.

Implicit in this formulation is the requirement that every feature point be visible in every frame. Handling occlusion for the orthographic factorization method has been covered in [Tomasi and Kanade, 1991b], and handling occlusion for the paraperspective method will be covered in a future paper.

3. The Orthographic Factorization Method

This section presents a summary of the orthographic factorization method. A more detailed description of the method can be found in [Tomasi and Kanade, 1991a, 1991b].

3.1. Orthographic Projection

The orthographic projection model assumes that all rays are projected from the object point parallel to the camera's optical axis so that they strike the image plane orthogonally, as illustrated in Figure 2.

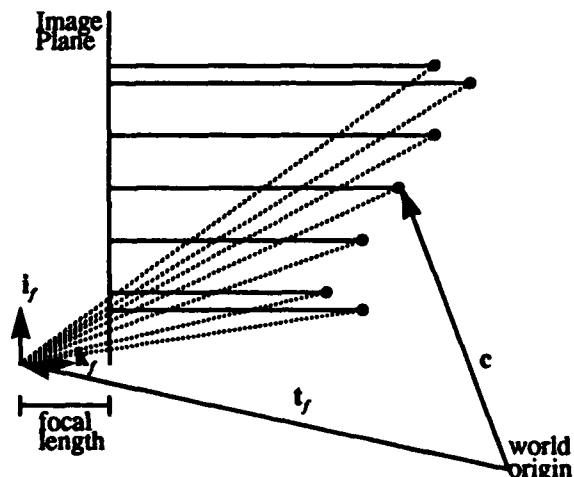


Figure 2. Orthographic projection in two dimensions

Dotted lines indicate true perspective projection

Under orthographic projection, a point p whose location is s_p will be observed in frame f at image coordinates (x_p, y_p)

$$x_p = i_f \cdot (s_p - t_f) \quad y_p = j_f \cdot (s_p - t_f) \quad (1)$$

We can rewrite these equations as

$$x_p = m_f \cdot s_p + c_x \quad y_p = n_f \cdot s_p + c_y \quad (2)$$

where

$$cx_f = -(t_f \cdot i_f) \quad cy_f = -(t_f \cdot j_f) \quad (3)$$

$$m_f = i_f \quad n_f = j_f \quad (4)$$

3.2. Decomposition

We organize all of the feature point coordinates (x_{fp}, y_{fp}) into a $2F \times P$ measurement matrix W .

$$W = \begin{bmatrix} x_{11} & \dots & x_{1P} \\ \dots & \dots & \dots \\ x_{F1} & \dots & x_{FP} \\ y_{11} & \dots & y_{1P} \\ \dots & \dots & \dots \\ y_{F1} & \dots & y_{FP} \end{bmatrix}. \quad (5)$$

Each column of the measurement matrix contains all the observations for a single point, while each row contains all the observed x-coordinates or y-coordinates for a single frame. We combine equation (2) for all points and frames into the matrix equation

$$W = MS + T \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}, \quad (6)$$

where M is the $2F \times 3$ motion matrix, S is the $3 \times P$ shape matrix, and T is a $2F \times 1$ translation vector.

Up to this point we have not put any restrictions on the location of the world origin, except that it be stationary with respect to the object. For simplicity, we set the world origin at the center-of-mass of the object, denoted by c , so that

$$c = \frac{1}{P} \sum_{p=1}^P s_p = 0. \quad (7)$$

This enables us to compute the i^{th} element of the translation vector T directly from W , simply as the average of the i^{th} row of the measurement matrix. We then subtract the translation from W , leaving us with a "registered" measurement matrix W' . Because W' is the product of a $2F \times 3$ motion matrix M and the $3 \times P$ shape matrix S , its rank is at most 3. We use singular value decomposition to factor W' into

$$W' = \hat{M} \hat{S}. \quad (8)$$

3.3. Normalization

The decomposition of equation (8) is not unique. In fact, any 3×3 non-singular matrix A and its inverse could be inserted between \hat{M} and \hat{S} , and their product would still equal W' . Thus the actual motion

and shape are given by

$$M = \hat{M}A \quad S = A^{-1}\hat{S}, \quad (9)$$

when the 3×3 invertible matrix A is selected appropriately. We observe that the motion matrix M must be of a certain form. Because i_f and j_f are unit vectors, we derive from equation (4) that

$$|m_f|^2 = 1 \quad |n_f|^2 = 1, \quad (10)$$

and because they are orthogonal,

$$m_f \cdot n_f = 0. \quad (11)$$

Equations (10) and (11) give us $3F$ equations which we call the *metric constraints*. Using these constraints, we solve for the 3×3 matrix A which, when multiplied by \hat{M} , produces the motion matrix M that best satisfies these constraints. Once the matrix A has been found, the shape and motion are computed from equation (9).

4. Paraperspective Factorization Method

4.1. Paraperspective Projection

In this paper, we use an approximation to perspective projection known as paraperspective projection, which was introduced by Ohta in order to solve a shape from texture problem. Paraperspective projection closely approximates perspective projection by modelling both the scaling effect and the position effect (so called because the amount of apparent rotation depends on the object's position in the image relative to the center of projection [Aloimonos, 1990]), while retaining the linear properties of orthographic projection. The paraperspective projection of an object onto an image, illustrated in Figure 3, involves two steps.

1. The points of the object are projected along the direction of the line connecting the focal point of the camera to the object's center-of-mass, onto a plane parallel to the image plane and passing through the object's center-of-mass.
2. These points are then projected onto the image plane using perspective projection. Because the points are all on a plane parallel to the image plane, this is equivalent to simply scaling the image by the ratio of the camera focal length and the distance between the two planes.¹

In general, the projection of a point p along direction r , onto the plane with normal n and distance from the origin d , is given by the equation $p' = p - \frac{p \cdot n - d}{r \cdot n} r$. We project the object point s_p

$$\mathbf{c} = \sum_{p=1}^P \mathbf{s}_p = 0. \quad (22)$$

Using this and equation (16) we can write

$$\begin{aligned} \sum_{p=1}^P x_{fp} &= \sum_{p=1}^P (\mathbf{m}_f \cdot \mathbf{s}_p + cx_f) = \mathbf{m}_f \cdot \sum_{p=1}^P \mathbf{s}_p + Pcx_f = Pcx_f \\ \sum_{p=1}^P y_{fp} &= \sum_{p=1}^P (\mathbf{n}_f \cdot \mathbf{s}_p + cy_f) = \mathbf{n}_f \cdot \sum_{p=1}^P \mathbf{s}_p + Pcy_f = Pcy_f \end{aligned} \quad (23)$$

Therefore we can compute cx_f and cy_f immediately from the image data as

$$cx_f = \frac{1}{P} \sum_{p=1}^P x_{fp} \quad cy_f = \frac{1}{P} \sum_{p=1}^P y_{fp}. \quad (24)$$

We subtract these values from the corresponding rows in W , giving the registered measurement matrix

$$W' = \begin{bmatrix} x_{11} & \dots & x_{1P} \\ \dots & \dots & \dots \\ x_{F1} & \dots & x_{FP} \\ y_{11} & \dots & y_{1P} \\ \dots & \dots & \dots \\ y_{F1} & \dots & y_{FP} \end{bmatrix} - \begin{bmatrix} cx_1 \\ \dots \\ cx_F \\ cy_1 \\ \dots \\ cy_F \end{bmatrix} \begin{bmatrix} 1 & \dots & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1 \\ \dots \\ \mathbf{m}_F \\ \mathbf{n}_1 \\ \dots \\ \mathbf{n}_F \end{bmatrix} \begin{bmatrix} s_1 & \dots & s_P \end{bmatrix} \quad (25)$$

Since W' is the product of two matrices each of rank at most 3, W' has rank at most 3, just as it did in the orthographic projection case. If there is noise present, the rank of W' will not be exactly 3, but by computing the singular value decomposition (SVD) of W' and only retaining the largest 3 singular values, we can factor W' into

$$W' = \hat{M}\hat{S}, \quad (26)$$

where \hat{M} is a $2F \times 3$ matrix and \hat{S} is a $3 \times P$ matrix. Using the SVD to perform this factorization guarantees that the product $\hat{M}\hat{S}$ is the best possible rank 3 approximation to W' , in the sense that it minimizes the sum of squares difference between corresponding elements of W' and $\hat{M}\hat{S}$.

4.3. Normalization

Just as in the orthographic case, the decomposition of W' into the product of \hat{M} and \hat{S} is not unique. We need to find the matrix A that gives the true shape and motion

$$M = \hat{M}A \quad S = A^{-1}\hat{S} \quad (27)$$

Again, we determine this matrix A by observing that the motion matrix M must be of a certain form. We take advantage of the fact that i_f and j_f are unit

vectors and are orthogonal. According to equation (19), we observe that

$$|\mathbf{m}_f|^2 = \frac{1 + cx_f^2}{z_f^2} \quad |\mathbf{n}_f|^2 = \frac{1 + cy_f^2}{z_f^2}. \quad (28)$$

We know the values of cx_f and cy_f from our initial registration step, but because we do not know the value of the depth z_f , we cannot impose individual constraints on the magnitudes of \mathbf{m}_f and \mathbf{n}_f as we did in the orthographic factorization method. However, from equation (28) we see that

$$\frac{1}{z_f^2} = \frac{|\mathbf{m}_f|^2}{1 + cx_f^2} = \frac{|\mathbf{n}_f|^2}{1 + cy_f^2}. \quad (29)$$

Therefore we adopt the following constraint on the magnitudes of \mathbf{m}_f and \mathbf{n}_f :

$$\frac{|\mathbf{m}_f|^2}{1 + cx_f^2} - \frac{|\mathbf{n}_f|^2}{1 + cy_f^2} = 0. \quad (30)$$

In the case of orthographic projection, one constraint on \mathbf{m}_f and \mathbf{n}_f was that they each have unit magnitude, as required by equation (10). In the above paraperspective version of those constraints, we simply require that their magnitudes be in a certain ratio.

There is also a constraint on the angle relationship of \mathbf{m}_f and \mathbf{n}_f . From the definition of \mathbf{m}_f and \mathbf{n}_f , we have

$$\mathbf{m}_f \cdot \mathbf{n}_f = \frac{cx_f cy_f}{z_f^2}. \quad (31)$$

The problem with this constraint is that, again, z_f is unknown. We could choose to use either value from equation (29) for $1/z_f^2$, since theoretically they should be equal, but we use the average of the two quantities. We choose the arithmetic mean over the geometric mean or some other measure in order to keep the constraints linear in $Q = A^T A$. Thus our second constraint becomes

$$\mathbf{m}_f \cdot \mathbf{n}_f - \frac{cx_f cy_f |\mathbf{m}_f|^2}{2(1 + cx_f^2)} - \frac{cx_f cy_f |\mathbf{n}_f|^2}{2(1 + cy_f^2)} = 0. \quad (32)$$

This is the paraperspective version of the orthographic constraint given by equation (11), which required that the dot product of \mathbf{m}_f and \mathbf{n}_f be zero.

Equations (30) and (32) are homogeneous constraints, which could be trivially satisfied by the solution $M = 0$. To avoid this solution, we impose the additional constraint that

$$|\mathbf{m}_1| = 1. \quad (33)$$

This does not effect the final solution except by a scaling factor.

Equations (30), (32), and (33) are the paraperspective version of the *metric constraints*, and we compute the 3×3 matrix A such that $M = \hat{M}A$ best satisfies the metric constraints in the least sum-of-squares error sense. This is a simple problem because we have been careful to ensure that they are linear constraints in the 6 unique elements of the symmetric 3×3 matrix $Q = A^T A$. We use the metric constraints to compute Q , compute its Jacobi Transformation $Q = L \Lambda L^T$, where Λ is the diagonal eigenvalue matrix, and as long as Q is positive definite, $A = (L \Lambda^{1/2})^T$.

4.4. Shape and Motion Recovery

Once the matrix A has been determined, we compute the shape matrix S and the motion matrix M using equation (27). For each frame f , however, there is a more complex relationship between the actual translation and rotation vectors and the \mathbf{m}_f and \mathbf{n}_f vectors, which are the rows of the matrix M . From equation (19) we can see that

$$\hat{\mathbf{i}}_f = z_f \mathbf{m}_f + c x_f \hat{\mathbf{k}}_f \quad \hat{\mathbf{j}}_f = z_f \mathbf{n}_f + c y_f \hat{\mathbf{k}}_f. \quad (34)$$

From this and the knowledge that $\hat{\mathbf{i}}_f$, $\hat{\mathbf{j}}_f$, and $\hat{\mathbf{k}}_f$ must be orthonormal, we determine that

$$\begin{aligned} \hat{\mathbf{i}}_f \times \hat{\mathbf{j}}_f &= (z_f \mathbf{m}_f + c x_f \hat{\mathbf{k}}_f) \times (z_f \mathbf{n}_f + c y_f \hat{\mathbf{k}}_f) = \hat{\mathbf{k}}_f \\ |\hat{\mathbf{i}}_f| &= |z_f \mathbf{m}_f + c x_f \hat{\mathbf{k}}_f| = 1 \\ |\hat{\mathbf{j}}_f| &= |z_f \mathbf{n}_f + c y_f \hat{\mathbf{k}}_f| = 1 \end{aligned} \quad (35)$$

Again, we do not know a value for z_f , but using the relations specified in equation (29) and the additional knowledge that $|\hat{\mathbf{k}}_f| = 1$, equation (35) can be reduced to

$$G_f \hat{\mathbf{k}}_f = H_f, \quad (36)$$

where

$$G_f = \begin{bmatrix} (\tilde{\mathbf{m}}_f \times \tilde{\mathbf{n}}_f) \\ \tilde{\mathbf{m}}_f \\ \tilde{\mathbf{n}}_f \end{bmatrix} \quad H_f = \begin{bmatrix} 1 \\ -c x_f \\ -c y_f \end{bmatrix} \quad (37)$$

$$\tilde{\mathbf{m}}_f = \sqrt{1 + c x_f^2} \frac{\mathbf{m}_f}{|\mathbf{m}_f|} \quad \tilde{\mathbf{n}}_f = \sqrt{1 + c y_f^2} \frac{\mathbf{n}_f}{|\mathbf{n}_f|} \quad (38)$$

We compute $\hat{\mathbf{k}}_f$ simply as

$$\hat{\mathbf{k}}_f = G_f^{-1} H_f \quad (39)$$

and then compute

$$\hat{\mathbf{i}}_f = \tilde{\mathbf{n}}_f \times \hat{\mathbf{k}}_f \quad \hat{\mathbf{j}}_f = \hat{\mathbf{k}}_f \times \tilde{\mathbf{m}}_f \quad (40)$$

There is no guarantee that the $\hat{\mathbf{i}}_f$ and $\hat{\mathbf{j}}_f$ given by this equation will be orthonormal, because \mathbf{m}_f and \mathbf{n}_f may not have exactly satisfied the metric constraints. Therefore we actually compute the orthonormal $\hat{\mathbf{i}}_f$ and $\hat{\mathbf{j}}_f$ which are closest to the values given by equation (40). Due to the arbitrary world coordinate orientation, to obtain a unique solution we then rotate the computed shape and motion to align the world axes with the first frame's camera axes, so that $\hat{\mathbf{i}}_1 = [1 \ 0 \ 0]^T$ and $\hat{\mathbf{j}}_1 = [0 \ 1 \ 0]^T$.

All that remain to be computed are the translations for each frame. We calculate the depth z_f from either part or some combination of the parts of equation (29). Since we already know $c x_f$, $c y_f$, $\hat{\mathbf{i}}_f$, and $\hat{\mathbf{j}}_f$, we can calculate $\hat{\mathbf{i}}_f$ using equations (17) and (18).

5. Experiments

5.1. Parameters

To test our method, we created synthetic point sequences using a perspective projection model of objects undergoing motion. We perturbed the coordinates of each point by adding gaussian noise, whose standard deviation was varied from 0 to 4 pixels (of a 512×512 pixel image). We used 3 different object shapes, each of unit size and containing approximately 60 feature points. All of the test runs consisted of 60 image frames of the object rotating through a total of 30 degrees each of roll, pitch, and yaw. The depth, representing the distance from the camera's focal point to the front of the object in the first frame, was varied from 3 to 60 times the object size. In generating our synthetic images, for each depth we chose the largest focal length which would keep the object in the field of view. For each combination of object, depth, and noise, we performed three tests, using different random noise each time.

We used several different methods to recover the shape and motion of the object and compared the accuracy of the results; the orthographic factorization method, the scaled orthographic factorization method (see [Poelman and Kanade, 1992]), the paraperspective factorization method, and the full perspective method which iteratively solves the perspective projection equations (see [Poelman and Kanade, 1992]). Because iterative methods are

generally very sensitive to initial conditions, we tested the latter method using two different starting values: the results of the paraperspective factorization method, and the true shape and motion. The last method is unfortunately not an option in real systems, but indicates what is essentially an upper bound on the accuracy achievable using a least sum-of-squares difference formulation of the full perspective projection model, without making assumptions about the motion or object shape.

5.2. Error Measurement

We present here the total shape error, rotation error, X-Y offset error, and Z offset (depth) error. The term "offset" refers to translations along the camera components; the X offset is $\hat{t}_x \cdot \hat{i}_r$, the Y offset is $\hat{t}_y \cdot \hat{j}_r$, and the Z offset is $\hat{t}_z \cdot \hat{k}_r$. The shape and translations are only determined up to scaling factor, since it's not possible to distinguish a house 50m away from a 1/10th scale model of the house 5m away. To compute the shape error, we find a scaling factor which minimizes the root-mean-square (RMS) error between the true and computed shape, and then return this error. We use the same method for the X-Y offset, and for the Z offset. The rotation error is computed as the RMS of the size in radians of the angle by which a computed camera frame must be rotated about some axis to produce the true camera frame.

5.3. Results

We found that the paraperspective method performed significantly better than the orthographic factorization method in image sequences in which there was depth translation or the object was not centered in the image. In the experiments in which the object was centered in the image and there was no depth translation, we found that the orthographic factorization method performed well, and the paraperspective factorization method provided no significant improvement. This is not surprising, since the orthographic method is in effect incorporating knowledge about the object motion - that the object is centered in the image and not translating toward or away from the camera.

The average error results were very similar for all of the objects, so our graphs show the average error over all 3 runs of all 3 objects. Figure 4 shows how the various methods performed in a scenario in which the object moved across the screen one unit horizontally and one unit vertically, and moved

away from the camera by a total of one half the object's initial distance from the camera (thus in a test case in which the object's depth in the first frame was 3.0, its depth in the last frame was 4.5). These tests were done using a noise standard deviation of 2 pixels, which we consider a rather high noise level. At low depths, perspective distortion is a significant source of error in the computed results. Interestingly, our experiments show that for objects farther from the camera than 7 times the object size, refining the paraperspective solution using the perspective iteration technique improves the rotation and translation very little. However, even at depths beyond 30 times the object size, the perspective refinement method significantly improves the shape.

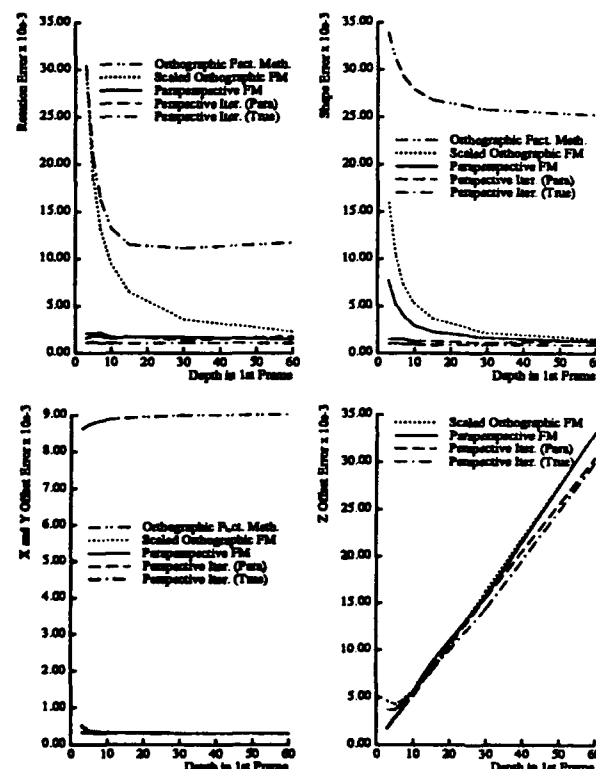


Figure 4. Methods compared for a typical case

The behavior of the paraperspective factorization method for the same motion scenario over a range of noise levels is shown in Figure 5. Once the object is far enough from the camera that perspective effects are minor, the error in the computed solution is nearly proportional to the amount of noise in the input.

We implemented the methods in C and performed the experiments on a Sun 4/65. Solving the system of 60 frames and 60 points required about 20-24 seconds for each of the three factorization meth-

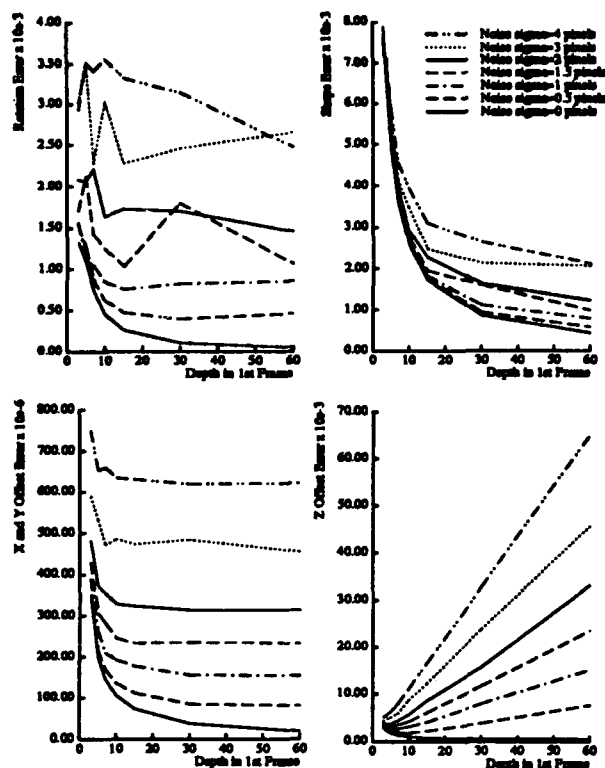


Figure 5. Paraperspective shape and motion recovery by noise level

ods, with much of this time spent computing the singular value decomposition of the measurement matrix. The perspective iteration method required about 250 seconds to solve the same system.

6. Conclusions

The principle that the measurement matrix has rank 3, as put forth by Tomasi and Kanade in [1991a], depended on the use of an orthographic projection model. We have shown in this paper that this important result also holds for the case of paraperspective projection, which closely approximates perspective projection, and have devised a paraperspective factorization method based on this model.

In general image sequences in which the object being viewed translates significantly toward or away from the camera or across the camera's field of view, the paraperspective factorization method performs significantly better than the orthographic method. In image sequences in which the object is close to the camera, the paraperspective factorization method still provides accurate motion results, and provides a good approximation of the object shape; this solution can be further refined using an iterative perspective method.

The paraperspective factorization method computes the distance from the camera to the object in each image, which enables its use in a wider range of scenarios. The method performs well over a wide range of motions, is efficient, and is robust.

References

- [Aloimonos, 1990] John Y. Aloimonos, *Perspective Approximations*, Image and Vision Computing, 8(3):177-192, August 1990.
- [Broida et al., 1990] T. Broida, S. Chandrashekar, and R. Chellappa, *Recursive 3-D Motion Estimation from a Monocular Image Sequence*, IEEE Transactions on Aerospace and Electronic Systems, 26(4):639-656, July 1990.
- [Mundy and Zisserman, 1992] Joseph L. Mundy and Andrew Zisserman, *Geometric Invariance in Computer Vision*, The MIT Press, 1992, p. 512.
- [Ohta et al., 1981] Yu-ichi Ohta, Kiyoshi Maenobu, and Toshiyuki Sakai, *Obtaining Surface Orientation from Texts Under Perspective Projection*, Proceedings of the 7th International Joint Conference on Artificial Intelligence, pp. 746-751, August 1981.
- [Poelman and Kanade, 1992] Conrad J. Poelman and Takeo Kanade, *A Paraperspective Factorization Method for Shape and Motion Recovery*, Technical Report CMU-CS-92-208, Carnegie Mellon University, Pittsburgh, PA, October 1992.
- [Press et al., 1988] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 1988.
- [Taylor et al., 1991] Camillo Taylor, David Kriegman, and P. Anandan, *Structure and Motion From Multiple Images: A Least Squares Approach*, IEEE Workshop on Visual Motion, pp. 242-248, October 1991.
- [Tomasi and Kanade, 1991a] Carlo Tomasi and Takeo Kanade, *Shape and Motion from Image Streams: a Factorization Method - 2. Point Features in 3D Motion*, Technical Report CMU-CS-91-105, Carnegie Mellon University, Pittsburgh, PA, January 1991.
- [Tomasi and Kanade, 1991b] Carlo Tomasi and Takeo Kanade, *Shape and Motion from Image Streams: a Factorization Method*, Technical Report CMU-CS-91-172, Carnegie Mellon University, Pittsburgh, PA, September 1991.
- [Tsai and Huang, 1984] Roger Tsai and Thomas Huang, *Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects with Curved Surfaces*, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6(1):13-27, January 1984.

Understanding Noise : The Critical Role of Motion Error in Scene Reconstruction *

J. Inigo Thomas Allen Hanson John Oliensis
Department of Computer Science, University of Massachusetts
Amherst, MA 01003

Abstract

In Structure from Motion algorithms, the error in the estimated motion affects each reconstructed 3D point in a systematic way. This paper attempts to isolate the effect of the motion error (as correlations in the structure error) and shows theoretically that these correlations can improve existing multi-frame Structure from Motion techniques. Finally it is shown that new experimental results and previously reported work confirm the theoretical predictions.

1 Introduction

Due to problems in two-frame Structure from Motion (SFM) [4], an obvious solution has been to use more than two frames to reconstruct the environment. Although it is theoretically conceivable that using enough different views should make it possible to achieve any required accuracy, achieving stable and reliable 3D reconstructions is still difficult. Based on an algorithm presented by Thomas and Oliensis [18] [12] we show that in order to obtain a stable and reliable reconstruction (for general motion), the effect of the interframe motion error has to be taken into account; we argue that ignoring this component could have resulted in the failure of previous (recursive) multi-frame SFM (MFSFM) algorithms. The theoretical and experimental evidence for the critical role of motion error (in MFSFM) is the main contribution of this paper.

2 Problems in MFSFM

Using multiple images introduces different sets of problems, depending on whether the algorithms are *batch* methods or *recursive* methods. Due to the large search spaces involved, batch methods impose restraints on the camera motion ([13] [2] [10] [5] [17]) or constrain the camera model [20]). However recursive MFSFM algorithms need not impose such restraints (although early research typically was constrained; cf. discussion in Section 4). Recursive MFSFM algorithms are also more practical for robot navigation applications since neither time nor storage is lost waiting until enough frames have

been acquired. However, in order to recursively refine the 3D structure, a reliable estimate of the *error* in the 3D structure is required. If the estimate of the error is unreliable, this results in random behavior or possibly systematically erroneous behavior. One of the biggest problems in MFSFM is that it is difficult to represent the error in the structure reliably.

One representation of the reconstruction error is a complete covariance matrix. That is, if the scene is reconstructed by n 3D points, then the reconstruction error is represented by a covariance matrix of size $9n^2$. This covariance matrix is difficult to compute, expensive to store, and computationally complex to manipulate. Presumably for these reasons, almost all of the work in recursive MFSFM has only used a portion of the covariance matrix, with poor results. It is being argued in [19] that, in general, every entry of the covariance matrix is meaningful; arbitrarily neglecting entries in the matrix could amount to a bad approximation of the actual reconstruction error (for general camera motion). A simplistic explanation is as follows. The main source of error in all structure from motion algorithms is the error in the estimated camera motion. The motion error affects all the 3D coordinates of the reconstruction in a systematic way; i.e., the errors in all the 3D coordinates are correlated. For example, if the translation component of the camera motion is erroneous, each 3D coordinate would be displaced along the same direction. Since every element of the covariance matrix represents the correlation of the error between pairs of 3D points, arbitrarily neglecting portions of the matrix (e.g. all off-diagonal terms) could have serious consequences. The following section is a theoretical analysis of the meaning and the role of cross-correlations in recursive MFSFM algorithms.

3 Theoretical Motivation for Using Cross-correlations

Let R be the entire reconstruction, made up of n 3D points (R_i , $i = 1 \dots n$).

Since each R_i is obtained from a two-frame algorithm effectively by triangulation, it has two sources of error. The first source is the error in the interframe motion, or the relative orientation of the cameras. The second source of error is the noise in the image coordinates. A

*This work was supported by DARPA (via TACOM) under contract number DAAE07-91-C-R035 and by NSF under grant CDA-8922572.

reasonable approximation of the total error in R_i is to express it (using first order terms) as the sum of the error due to the interframe motion and the error due to the image coordinates; the error in R_i is

$$dR_i = \frac{\delta R_i}{\delta W} dW + \frac{\delta R_i}{\delta I_i} dI_i \quad (1)$$

where W represents the interframe motion and I_i represents the image coordinates of the point; dW and dI_i represent the respective errors.

When the error in R is represented as a covariance matrix the elements of this matrix are given by the following equation:

$$COV(dR dR^T) = \begin{pmatrix} E(dR_1 dR_1) & \dots & E(dR_1 dR_n) \\ & \ddots & \\ E(dR_n dR_1) & \dots & E(dR_n dR_n) \end{pmatrix} \quad (2)$$

where $E(x)$ denotes the expected value of x .

In this theoretical analysis, in order to bring out the meaning and role of the cross-correlation terms clearly, we will assume that we have a reconstruction consisting of just two points.

3.1 The Meaning of Cross-correlations: The Two Point Case

In this case, the covariance matrix is reduced to

$$COV(dR dR^T) = \begin{pmatrix} E(dR_1 dR_1) & E(dR_1 dR_2) \\ E(dR_2 dR_1) & E(dR_2 dR_2) \end{pmatrix} \quad (3)$$

This covariance matrix has four correlation terms, two of which are equivalent ($E(dR_1 dR_2)$ and $E(dR_2 dR_1)$). The other two ($E(dR_1 dR_1)$ and $E(dR_2 dR_2)$) are the covariance of the error in R_1 and R_2 ; these are typically assumed to represent the complete error. However, here we will concentrate on the cross-correlation term, $E(dR_1 dR_2)$.

Using Equation 1 the cross-correlation term can be expanded as in Equation 4:

$$E(dR_1 dR_2) = E\left[\left(\frac{\delta R_1}{\delta W} dW + \frac{\delta R_1}{\delta I_1} dI_1\right)\left(\frac{\delta R_2}{\delta W} dW + \frac{\delta R_2}{\delta I_2} dI_2\right)^T\right] \quad (4)$$

Since it is realistic to assume that any two arbitrary image coordinates (of chosen points) are corrupted by independent noise, i.e.

$$E(dI_1 dI_2) = 0 \quad (5)$$

one of the terms in the expansion of Equation 4 will vanish. The resultant expansion is given in Equation 6:

$$\begin{aligned} E(dR_1 dR_2) &= \frac{\delta R_1}{\delta W} E(dW dW^T) \frac{\delta R_2}{\delta W} \\ &+ \frac{\delta R_1}{\delta W} E(dW dI_2^T) \frac{\delta R_2}{\delta I_2} + \frac{\delta R_1}{\delta I_1} E(dI_1^T dW) \frac{\delta R_2}{\delta W} \end{aligned} \quad (6)$$

Given Equation 6, the only way the cross-correlation term will end up being zero is when the three terms fortuitously cancel; in all other cases the cross-correlation term has an effect on the performance of the recursive MFSFM algorithm. Furthermore, the situations in which the three terms cancel each other out are most likely rare.

For the sake of exposition let us assume that the coordinates of the two points have changed considerably between the two images, resulting in large optical flow. Therefore, a small error in the optical flow (which corresponds to a small error in I) has little effect on the error in the motion, dW ; i.e.

$$E(dW dI_i^T) \approx 0 \quad i = 1, 2 \quad (7)$$

For this particular case, the expansion of Equation 6 is:

$$E(dR_1 dR_2) = \frac{\delta R_1}{\delta W} COV(dW dW) \frac{\delta R_2}{\delta W} \quad (8)$$

Equation 8 shows that the cross-correlation is directly proportional to the motion error, represented as the covariance of the error in the motion (dW). If Equation 7 does not hold the situation is more complicated: the cross-correlation is influenced not only by the motion error but also (indirectly) by the error in the image coordinates. In either case, the cross-correlation term is closely related to the motion error.

3.2 The Effect of Cross-correlations in Kalman Filtering

In this section the analysis is extended to study the effect of cross-correlations on refining reconstructions using the Kalman filter.

The goal of the Kalman filter is to optimally fuse the reconstructions over time and obtain the best reconstruction (by limiting the reconstruction error). If we assume that the noise in every new reconstruction ($R(t)$ at time t) is Gaussian, then the optimal fused reconstruction is the sum of the individual reconstructions weighted by the inverse of their covariances. Given this the optimal fused reconstruction (\tilde{R}) at time t is as follows (i.e. standard Kalman filtering [6]):¹

$$\tilde{R}(t) = N \sum_{i=1}^t COV(R(t))^{-1} R(t) \quad (9)$$

In order to determine the exact contribution of a single reconstruction ($COV(R)^{-1} R$ or R^W , for weighted R) at any time (t) the covariance can be expanded using Equation 3 (and assuming Equation 7 is valid) in the following way:

$$COV(R) = \begin{pmatrix} S_1 + M_{11} & M_{12} \\ M_{21} & S_2 + M_{22} \end{pmatrix} \quad (10)$$

where

$$S_i = \frac{\delta R_i}{\delta I_i} COV(dI_i dI_i) \frac{\delta R_i}{\delta I_i} \quad (11)$$

and

$$M_{ij} = \frac{\delta R_i}{\delta W} COV(dW dW) \frac{\delta R_j}{\delta W} \quad (12)$$

S_i represents the error in the 3D coordinates due to the error in the image coordinates (dI_i) assuming that the motion is perfectly known; M_{ij} represents the error in the 3D coordinates due to the error in the interframe

¹ N (in Equation 9) is a normalising term which is irrelevant for this analysis.

camera motion (dW) assuming that the image coordinates are perfectly known.

The *weighted* R can now be written as:

$$R^W = \begin{pmatrix} S_1 + M_{11} & M_{12} \\ M_{21} & S_2 + M_{22} \end{pmatrix}^{-1} \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} \quad (13)$$

Equation 13 can be expanded after Bar-Shalom and Fortmann [14]. In this analysis, let us now concentrate on the effect of the cross-correlation on a single optimally fused 3D coordinate (\bar{R}_1); all of the relevant information is contained in the first row in the expansion of Equation 13, which is:

$$R_{row1}^W = S_{11} + M_{11} - M_{12}(S_{22} + M_{22})^{-1}M_{12}^T - 1 \\ (R_1 - M_{12}(S_{22} + M_{22})^{-1}R_2) \quad (14)$$

The second term (in Equation 14) can be thought of as a *Corrected* R_1 :

$$Corrected R_1 = R_1 - M_{12}(S_{22} + M_{22})^{-1}R_2 \quad (15)$$

If there is no error in the motion - i.e. M_{12} is zero - the *Corrected* R_1 is identical to R_1 . However, since this is generally not true in practice, the value of R_2 has a corrective effect on R_1 . The magnitude of the correction depends on the size of the cross-correlation M_{12} . Since we have shown that the cross-correlation captures the motion error (cf. Section 3.1), the magnitude of the correction depends on the (shared) motion error that corrupts both R_1 and R_2 .

The covariance of *Corrected* R_1 is

$$COV(Corrected R_1) = E([R_1 - M_{12}(S_{22} + M_{22})^{-1}R_2]^2) \quad (16)$$

Again, this can be simplified to obtain:

$$COV(Corrected R_1) = S_{11} + M_{11} - M_{12}(S_{22} + M_{22})^{-1}M_{12}^T \quad (17)$$

As stipulated by Kalman filtering, any contribution (towards the fused optimal estimate) has to be weighted by the inverse of its covariance. Thus we expect that *Corrected* R_1 (Equation 15) should be weighted by the inverse of its covariance. Since the right-hand side of Equation 17 turns out to be equal to the first term of Equation 14 above, this is *exactly* the case.

This analysis reveals that the cross-correlation terms are important. If the interframe motion error is large, then the cross-correlation terms become significant and play a crucial role. Since in SFM the motion error is typically large [4] we predict that without cross-correlations the benefits of Kalman filtering are lost, i.e. the fused reconstruction would be neither stable nor accurate. In the next section we present experimental evidence to this effect.

4 Experimental Data

The previously reported MFSFM algorithms conform to the prediction of the last section. Heel [7] approximates the entire covariance matrix by just the error terms relating to one coordinate Z (i.e., when reconstructing n 3D points, his covariance matrix has n elements rather than

the full $9n^2$ elements); only qualitative results are reported and the camera motion is restricted to a straight line. Matthies et. al. [11] also use only n elements to represent the reconstruction error. However, in their experiment the camera motion is known accurately, in which case the cross-correlations should *not* play a role; their reconstruction is within 0.5% error using 11 images. Shigang, Tsuji, and Imai [15] also use only n terms to approximate their error, but consider more general motions than Heel. When they allow the camera to move freely in a plane, their reconstruction error is 15% even with as many as 40 images. Ando [1] also uses n elements (for general camera motion) but only simulation experiments are reported.

The next category of approximations involve using $9n$ elements to approximate the $9n^2$ covariance matrix. Stephens et al. [16] report reconstructions within 1% error for 1 point after 50 frames in the case of motion straight ahead. Cui, Weng and Cohen [3] also use $9n$ elements to approximate the full covariance matrix and apply the algorithm for the case of general camera motion. The reported accuracy of the reconstruction (from a real image sequence) fluctuates randomly. Since no comparison with the ground truth is reported it is unclear as to how well this algorithm really does.

The algorithm developed by Thomas and Oliensis [18] [12] is the only recursive MFSFM algorithm (for general motion) that uses the full covariance matrix with $9n^2$ elements. Apart from using cross-correlations, their algorithm is similar to previous recursive (Kalman filter) MFSFM algorithms. Highly accurate reconstructions (as accurate as the ground truth) have already been reported by Thomas and Oliensis [18] for image sequences with no constraints on the robot camera motion. Here, their algorithm is used to test for the effect of cross-correlations in real image sequences by comparing results from the same algorithm with and without cross-correlations. Such a comparison has not been previously done; it will be presented in the following section for two real image sequences.²

4.1 Experiment I: Reconstruction of a Rotating Box

A box was rotated in steps of approximately 4 degrees around its vertical axis, and nine images were obtained by a stationary camera (cf. Fig 1). The camera parameters are: focal length 6cm, fov (23.4°, 22.4°), and image size 256 × 242 pixels. The available ground truth had an accuracy of 1.5mm. In this experiment 35 corners of the white squares on the box were chosen to be reconstructed; the corners were tracked using the algorithm of Williams and Hanson [23]. Due to the well known scale ambiguity in SFM [21], the scale of the reconstruction was input as a single number at the beginning of the process.

In order to determine how well the shape of the box is reconstructed, each reconstruction is rotated and translated (rigidly) to align with the ground truth. The

²Thanks to Harpreet Sawhney and Rakesh Kumar for these sequences and the ground truth measurements.

mismatch between the aligned reconstruction and the ground truth is the error in the shape. The alignment that minimizes the mismatch error can be determined exactly (in closed form) by Horn's absolute orientation algorithm [8]. The overall error of the entire reconstruction is reported as an average of the individual mismatch errors over the set of reconstructed points.

The performance of the algorithm with and without the cross-correlation terms is presented here. For comparison we include the results from a standard two-frame approach: Horn's relative orientation algorithm [9].³

From the graph (in Fig. 2) it can be observed that the error in the two-frame reconstruction is fairly high (the average error is 8.8 mm; the dimensions of the box are 133 mm x 157 mm x 70 mm and the distance between any two points ranges from 15 mm to 207.19 mm). The random and high fluctuations (e.g. in frame 4 and frame 7) make the two-frame reconstructions unreliable. In the case when cross-correlations of the error are ignored in the multi-frame algorithm developed in [18], we can see that after an initial drop in error, the error fluctuates around 11 mm, but has a very slow decrease. Notice also that in frames 4 and 7 the error increases, showing that the algorithm is unable to ignore the erroneous individual two-frame reconstructions. However, when cross-correlations are not ignored, the average reconstruction error falls monotonically and remains as low as the error in the ground truth (1.5 mm) for the last 4 frames. Note that the final reconstruction (after 9 frames) of the MFSFM approach without cross-correlations is 6 times more erroneous than the final reconstruction when cross-correlations are considered.

4.2 Experiment II: Reconstruction of the Computer Science Lobby

A sequence of ten pictures were taken of the Computer Science lobby by a camera mounted on a moving Denning mobile robot moving almost straight ahead (Fig. 3). The camera parameters are: focal length 16mm, fov (29.3°, 22.9°), and image size 256 x 242 pixels. For this experiment 29 points were selected from the first image, making sure that each point was visible in the rest of the images. Point tracking was done as in the previous experiment. The error in the reconstruction is plotted in Fig. 4. In this case we are interested in reconstructing the structure of the scene, and not merely the shape. The error in the reconstructed 3D coordinates is reported as a percentage of the distance of the true 3D coordinates from the camera, averaged over the set of reconstructed points.

From the graph (Fig. 4) it can be observed that the error in the two-frame reconstructions is high and fluctuates randomly (around 8%). Also, from the graph we can see that the MFSFM approach that ignores cross-correlations leads to better accuracies than individual two-frame results. However, the reconstruction error does not decrease monotonically; instead it fluctuates around an error of 3.5 %.

³Horn's algorithm provides the input for Thomas and Oliensis' MFSFM algorithm.

Again, using cross-correlations yields the best accuracy of the three approaches compared here. Fig. 4 shows that the average reconstruction error falls almost monotonically, with a final error of 2.16% after ten frames. The final reconstruction (after ten frames) of the same MFSFM algorithm which ignores cross-correlations is 65% more erroneous than the final reconstruction which uses cross-correlations.

5 Conclusion

We have argued that the cross-correlation terms capture the interframe motion error and account for it. Ignoring the cross-correlations seem to have direct consequences on the accuracy and usefulness of the reconstructed models of the environment.

Although the cross-correlations have presumably been ignored because of their computational complexity, we have shown that they are crucial enough to warrant an attempt to make using them computationally feasible. Since the bottleneck of including cross-correlations is the time required to invert large matrices, one solution is a straightforward parallel implementation of the algorithm on a SIMD parallel machine such as the Image Understanding Architecture [22]. Future research will also be directed towards discovering other ways of reducing computational time such as using smaller (intersecting) subsets of points which are yet large enough to capture the underlying motion error.

References

- [1] H. Ando, "Dynamic Reconstruction of 3D Surfaces and 3D Motion," *Proc. IEEE workshop on visual motion*, Princeton, NJ, pp. 101-110, 1991.
- [2] T. J. Brodia and R. Chellappa, "Estimating the Kinematics and Structure of a Rigid Object from a Sequence of Monocular Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 497-513, 1991.
- [3] N. Cui, J. Weng and P. Cohen, "Extended Structure and Motion Analysis from Monocular Image Sequences," *Proceedings 3rd IEEE International Conference on Computer Vision*, Osaka, Japan, 1990, pp. 222-229.
- [4] R. Dutta and M. Snyder, "Robustness of Correspondence Based Structure from Motion," *Proceedings 3rd IEEE International Conference on Computer Vision*, Osaka, Japan, Dec. 1990.
- [5] W.O. Fransen, "Structure and Motion from Uniform 3-D Acceleration," *Proc. IEEE workshop on visual motion*, Princeton, NJ, pp. 14-20, 1991.
- [6] Technical Staff, The Analytical Sciences Corp., and A. Gelb, ed., *Applied Optimal Estimation*, MIT Press, 1986.
- [7] J. Heel, "Temporal Surface Reconstruction," *CVPR*, Hawaii, June, 1991, pp. 607-612.
- [8] B. K. P. Horn, "Closed Form Solution of Absolute Orientation Using Unit Quaternions," *J. Opt. Soc. Am. A*, vol. 4, pp. 629-642, 1987.
- [9] B. K. P. Horn, "Relative Orientation," *International Journal of Computer Vision*, Vol. 4, pp. 59-78, 1990.
- [10] R. V. R. Kumar, A. Tirmalai and R.C. Jain, "A Nonlinear Optimization Algorithm for the Estimation of Structure and Motion Parameters," *CVPR*, San Diego, CA, pp. 136-143, 1989.

- [11] L. Matthies, T. Kanade, and R. Szeliski, "Kalman Filter-Based Algorithms for Estimating Depth from Image Sequences," *International Journal of Computer Vision*, vol 3, pp. 209-236, 1989.
- [12] J. Oliensis and J. I. Thomas, "Incorporating Motion Error in Multi-frame Structure from Motion," *Proceedings IEEE Workshop on Visual Motion*, Princeton, pp 8-13, 1991.
- [13] H. S. Sawhney, J. Oliensis, and A. R. Hanson, "Description and Reconstruction from Image Trajectories of Rotational Motion", in *ICCV*, Osaka, Japan, December, 1990, pp. 494-498.
- [14] Y.Bar-Shalom and T.E.Fortmann, *Tracking and Data Association*, Academic Press, Orlando, Fl, 1991, pp. 277.
- [15] L.Shigang, S.Tsuji and M. Imai, "Determining of Camera Rotation from Vanishing Points of Lines on Horizontal Planes," *Proceedings 3rd IEEE International Conference on Computer Vision*, Osaka, Japan, 1990, pp. 499-502.
- [16] M.J.Stephens, R.J.Blissett, D.Charnley, E.P.Sparks and J.M.Pike, "Outdoor Vehicle Navigation Using Passive 3D Vision," *CVPR*, San Diego, CA, pp. 556-562, 1989.
- [17] C.J.Taylor, D.J.Kreigman, and P.Anandan, "Structure and Motion in Two Dimensions from Multiple Images: A Least Squares Approach," *Proc. IEEE workshop on visual motion*, Princeton, NJ, pp. 242-248, 1991.
- [18] J. Inigo Thomas and J. Oliensis, "Recursive Structure from Multi-frame Motion," *Proc. Darpa Image Understanding workshop*, San Diego, CA, 1992.
- [19] J. Inigo Thomas, *Scene Reconstruction Using an Image Sequence*, (in progress) Ph.D. Dissertation, University of Massachusetts, Amherst.
- [20] C. Tomasi and T. Kanade, "Factoring Image Sequences into Shape and Motion," *Proc. IEEE workshop on visual motion*, Princeton, NJ, pp. 21-28, 1991.
- [21] R.Y.Tsai and T.S.Huang, "Uniqueness and estimation of 3-D motion parameters and surface structures of rigid objects," pp. 135-171.
- [22] C.C.Weems, S.P.Levitan, A.R.Hanson, E.R. Riseman, D.B.Shu and J.G.Nash, "The Image Understanding Architecture," *IJCV*, 2, pp. 251-282.
- [23] L. R. Williams and A. R. Hanson, "Translating Optical Flow into Token Matches and Depth from Looming," *Proc. 2nd Intl. Conf. on Computer Vision*, pp. 441-448, 1988.

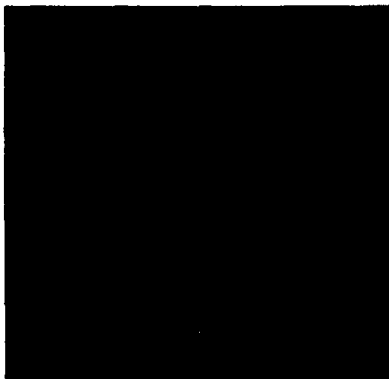


Fig. 1: Box Sequence

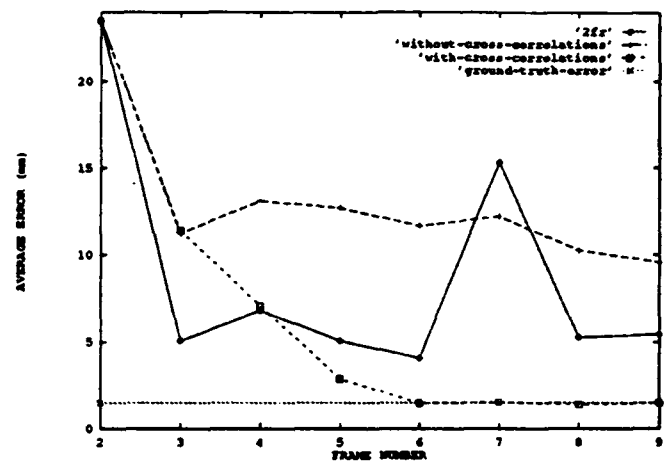


Fig. 2: Box Shape Error



Fig. 3: Lobby Sequence

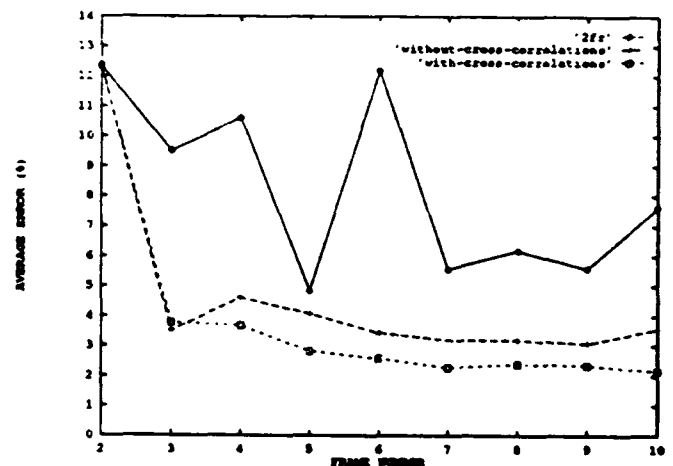


Fig. 4: Lobby Reconstruction Error

Translational Decomposition of Flow Fields*

Daryl T. Lawton and Warren F. Gardner

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

Abstract

We introduce a low-level description of image motion called the **local translational decomposition (LTD)**. This description associates with image features or small image areas, a three-dimensional unit vector describing the direction of motion of the corresponding environmental feature or small surface area. The local translational decomposition is derived by applying a procedure for processing purely translational motion to small overlapping image areas. This intermediate representation of motion considerably simplifies the inference of motion parameters for ego-motion and can support qualitative inferences for non-rigid motions. We first show how to compute the LTD from optic flow fields and then show how the LTD is used to recover the parameters of rigid body motions. We present three cases for which the recovery of motion parameters is particularly robust: motion constrained to a determined plane (the normal to the plane is known); motion constrained to an undetermined plane (the normal to the plane is not known); arbitrary motion relative to locally planar surfaces. We then discuss techniques for computing the local translational decomposition directly from real image sequences without the initial extraction of optic flow and other areas for future work.

1 Introduction

In previous work [Lawton, 1982], we developed a technique to process relative translational motion of a sensor with respect to a stationary environment or independently translating objects. This and related algorithms [Burger and Bhanu, 1989; Jain, 1983] are based on the strong geometric constraints on image motion in the case of translation - radial motion of image features from a focus of expansion (or contraction) determined by the intersection of the axis of translation with an imaging

surface [Gibson, 1950; Lee, 1980]. The technique presented in [Lawton, 1982] was based on optimizing a measure which described the quality of feature matches restricted to lie along the radial flow paths associated with a potential axis of translation. The optimization process involved searching over the surface of a unit sphere where each point corresponded directly to a possible direction of translation. The optimization combined the determination of the direction of translation and the corresponding image displacements into a single, mutually constraining computation. It was possible to determine the direction of translation to within a few degrees in small image areas using a few distinctive features.

In this paper we extend the translational processing algorithm to work with general rigid body and other cases of motion by applying the translational procedure to local portions of a flow field. This processing associates a direction of relative environmental motion with a local portion of a flow field and also an error measure reflecting the validity of the translational approximation. We call this description of image motion the **local translational decomposition (LTD)**. Computing the LTD begins by decomposing a flow field into small overlapping neighborhoods and then approximating the motion for each neighborhood as being produced by translational motion of the corresponding portion of the environment. This approximation associates a unit vector describing the direction of environmental motion with local portions of a flow field. Each unit vector has an associated fit-value reflecting the validity of the translational approximation.

The LTD is a low level representation of environmental motion which considerably simplifies the recovery of the sensor motion parameters. The local directions of motion and corresponding error measures are used as constraints to determine the actual parameters of motion and to recover the structure and layout of environmental surfaces. This is broken into four cases. For motion constrained to a plane of a known orientation (See Section 2.1), the local translational approximation is recovered directly from the intersection of flow vectors with the horizon line determined by the plane of motion. For motion constrained to a plane of unknown orientation (See Section 2.2), all of the computed LTD vectors must be perpendicular to the normal of the unknown plane. This constraint leads to a direct fitting procedure to recover

*This research is supported by the Advanced Research Projects Agency of the Department of Defense and is monitored by the U. S. Army Topographic Engineering Center under contract No. DACA76-92-C-0016

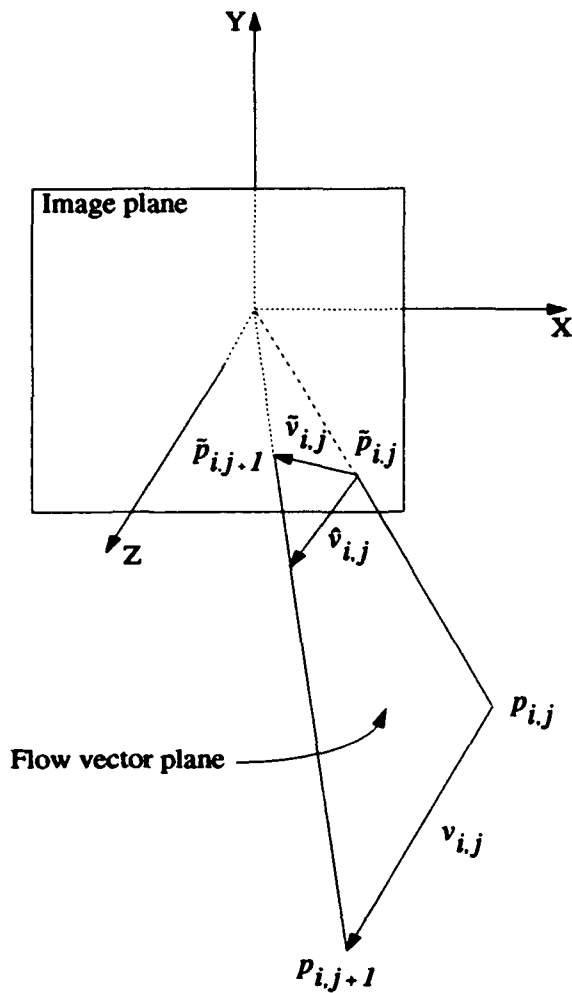


Figure 1: Camera coordinate system

the plane of motion. For motion relative to locally planar surfaces (see Section 2.3), the combination of local planarity and rigidity is used. For arbitrary motion, rigidity between environmental points is used to recover motion parameters from a small number of image locations (See Section 2 and Section 3.1).

The remainder of this section introduces the notation used throughout this paper. Section 2 describes how the local direction of translation is estimated from a flow field and cases of motion for which this is particularly robust. Section 3 describes how the parameters of relative sensor motion can be recovered from the estimated local directions of translation. Section 4 discusses computing the local translational decomposition directly from real image sequences without the initial extraction of optic flow and other areas for future work.

1.1 Notation

The coordinate system used in this paper is shown in Figure 1. The origin of this right-handed coordinate system lies at the focal point of the camera. The image plane is parallel to the xy -plane and is centered on the point $(0, 0, f)$, where f is the focal length of the camera. A

three-dimensional environmental point will be referred to as $p_{i,j} = (x_{i,j}, y_{i,j}, z_{i,j})$. The corresponding image point is $\bar{p}_{i,j} = (\bar{x}_{i,j}, \bar{y}_{i,j})$. The first subscript i is used to differentiate between points. The second subscript denotes the time interval. Thus, $p_{i,j}$ refers to the i th point at time j . A three-dimensional displacement which transforms $p_{i,j}$ into $p_{i,j+1}$ forms a vector. This vector will be referred to as $v_{i,j}$. The corresponding optic flow vector on the image plane is $\bar{v}_{i,j}$. In Section 2 a method for estimating $v_{i,j}$ is presented. This estimated vector will be referred to as $\hat{v}_{i,j}$. If $\hat{v}_{i,j}$ is correct, it will be parallel to $v_{i,j}$, but its depth will be unknown. $\hat{v}_{i,j}$ can be positioned anywhere along the rays of projection which pass through $\bar{p}_{i,j}$ and $\bar{p}_{i,j+1}$. Unless specified otherwise, $\hat{v}_{i,j}$ will be positioned at the image plane.

The motion of the camera can be described by six parameters. Let $r = (r_x, r_y, r_z)$ denote the axis of rotation, and $t = (t_x, t_y, t_z)$ the direction of translation. We assume the axis of rotation passes through the origin of the camera coordinate system. The magnitude of r is equal to the angle of rotation, and t is a unit vector.

2 Estimating Local Translation

In this section we show how to determine an axis of translation consistent with a local portion of a computed flow field. In section 4 we briefly discuss how to compute this directly from textured images without the initial extraction of a flow field.

Figure 1 shows that the plane formed by a flow vector and the focal point of the camera must include the estimated local translation vector (we refer to this as the *flow-vector plane* for a given flow vector). In the case of purely translational motion, the estimated local translation vector will be the same for all flow vectors in the neighborhood. Therefore, the estimated local translation vector is the vector which is parallel to all of the flow vector planes in the neighborhood. This observation leads directly to a method of solving for the estimated local translation.

The plane formed by $\hat{v}_{i,j}$ and the focal point of the camera must include $\hat{v}_{i,j}$. Let this plane be designated by its normal $n_{i,j}$.

$$n_{i,j} = \bar{p}_{i,j} \times \bar{p}_{i,j+1} \quad (1)$$

Since $n_{i,j}$ is perpendicular to $\hat{v}_{i,j}$

$$n_{i,j} \cdot \hat{v}_{i,j} = 0 \quad (2)$$

In the case of purely translational motion, the direction of $\hat{v}_{i,j}$ is constant for all i . Therefore, Equation 2 can be rewritten as

$$n_{i,j} \cdot \hat{v}_j = 0 \quad (3)$$

where $\hat{v}_j = \hat{v}_{i,j}$ for all i . This equation is linear with three unknowns, and can be solved using a least squares technique.

An error measure is used to evaluate the validity of the local translation approximation. The error measure we use is the average, taken over the local neighborhood, of the angle between each flow vector plane and the local translation. Using the normals $n_{i,j}$ from Equation 1, the

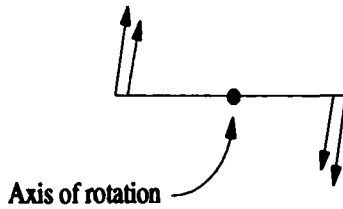


Figure 2: Local translation associated with a rotating line

error measure is defined as

$$\frac{1}{m} \sum_{i=1}^m \left| \sin^{-1} \left(\frac{n_{i,j} \cdot \hat{v}_j}{\|n_{i,j}\| \|\hat{v}_j\|} \right) \right| \quad (4)$$

where m is the number of flow vectors in the local neighborhood. Alternatively (and with greater expense), this measure could be optimized directly by a search procedure to determine an axis of translation.

In general, $\hat{v}_{i,j}$ is not constant for all i . However, in local areas $\hat{v}_{i,j}$ is approximately constant. For example, in Figure 2, points which are nearby on a line segment are shown to have approximately the same local translations when the line is rotated about its midpoint. Points near the axis of rotation would not have a good translational approximation as would be reflected in the corresponding error measure. Note that if the motion is composed of both a rotation and translation, the approximation will also be effected by environmental points at different depths, especially at occlusion boundaries. Since the flow vectors in the area of an occlusion boundary will not consistently emanate from a focus of expansion, the error measure given in Equation 4 returns a high value in these areas. Using the error measure, the unreliable occlusion areas can be avoided when computing the parameters of motion. Figure 3 shows the flow field for a scene containing multiple depths and undergoing an arbitrary motion. The error function derived from this flow field is shown in Figure 4. The scene contains two planes which occlude a planar background as well as each other. The planes, as well as the background, are skewed with respect to the image plane (i.e. the planes are receding in depth). The locations of the occlusion boundaries are obvious from the figure.

The method of LTD estimation discussed above was tested on several synthetic optic flow fields like the one shown in Figure 5. This flow field is the result of a rotation of 5.73° about the axis $(5, 4, 1)$, followed by a translation of $(100, 25, -75)$. All units are given in pixels. The field of view of the camera is 90° in both the X and Y directions. The image is 63×63 , and the focal length is 31. The rectangle overlay on the flow field represents the neighborhood over which the translational approximation is performed. The actual angles between the correct local translational vectors and the approximated local translational vectors at each position in the flow field is shown in Figure 6. The computed error measure based upon Equation 4 is shown as a surface plot in Figure 7. Notice that the computed error measure

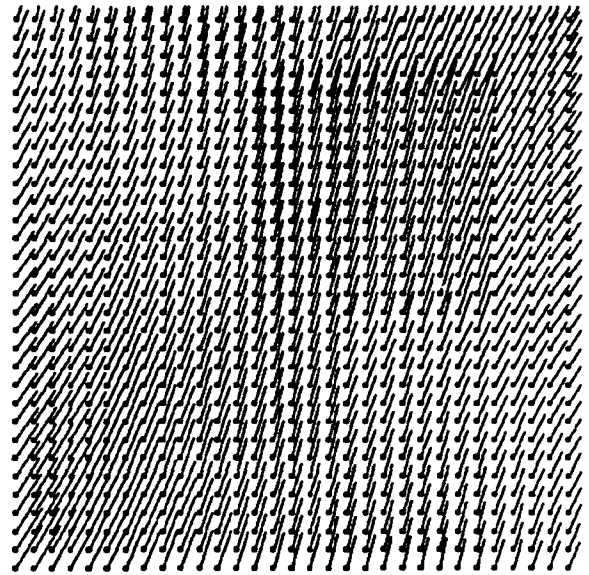


Figure 3: Flow field for an image containing occlusion

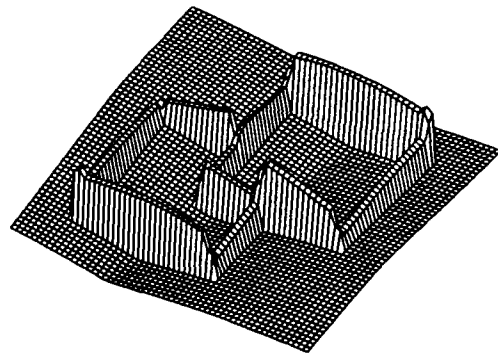


Figure 4: Error function for an image containing occlusion

in Figure 7 reflects a strong correspondence between the approximated translational vectors with the least error and the correct translational axes. This correspondence has been found to be typical. Figure 8 (a)-(c) shows the correct local directions of translation with the values of each component displayed as separate intensity plots. Since the translational vectors are represented as three-dimensional unit vectors with each component in the range of -1.0 to 1.0, Figure 8 displays the x, y, and z components of the local translation vectors with pure white corresponding to the value of 1.0 and pure black corresponding to -1.0. Figure 8 (d)-(f) shows the local translational values that were derived from the optic flow field using the approximation procedure. The

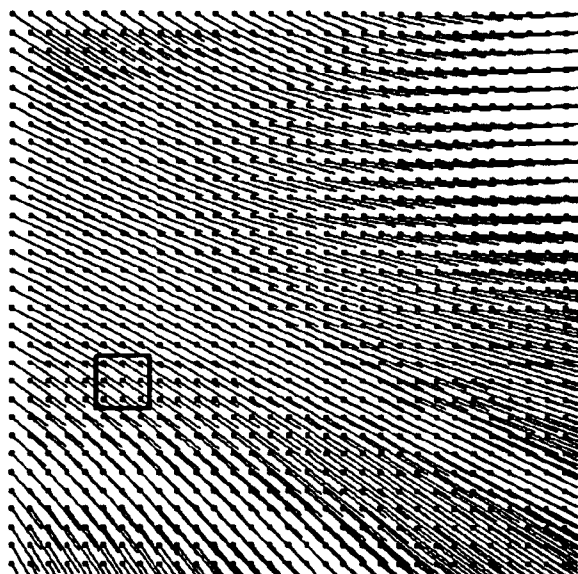


Figure 5: Optic flow field for a rotation of 5.73° about the axis $(5, 4, 1)$, translation of $(100, 25, -75)$

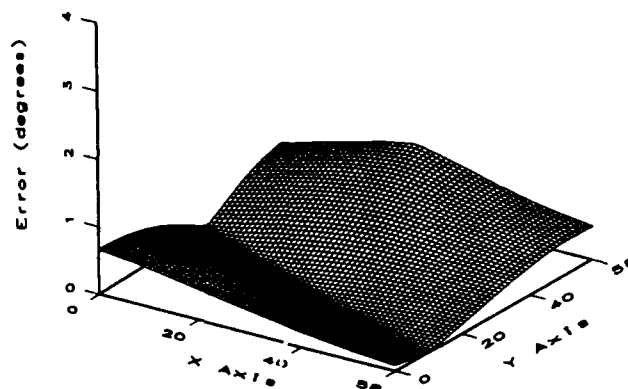


Figure 7: Evaluated error measure for flow in figure 5

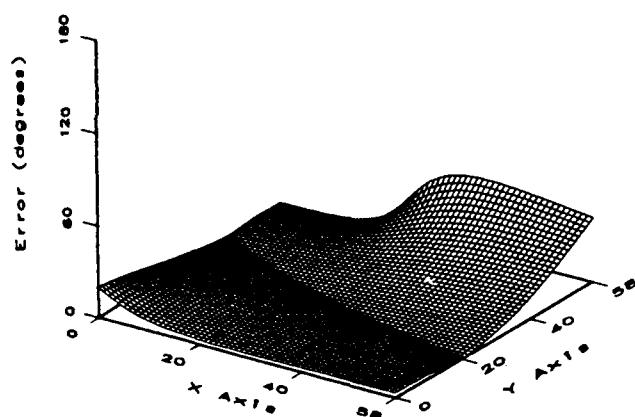


Figure 6: Actual errors for flow in figure 5

derived LTD vector components have been thresholded using the error measure given in Equation 4, so that only the best values are shown. These are then used for inferring the overall parameters of motion. The corresponding areas removed by the thresholding are shown by the enclosed white regions which contain a T.

2.1 Motion Constrained to a Determined Plane

It is particularly simple to recover the local translation from flow fields produced by environmental motion constrained to a determined plane (the normal to the plane is known). In this case, the environmental displacement vector $v_{i,j}$ must be perpendicular to the normal of the

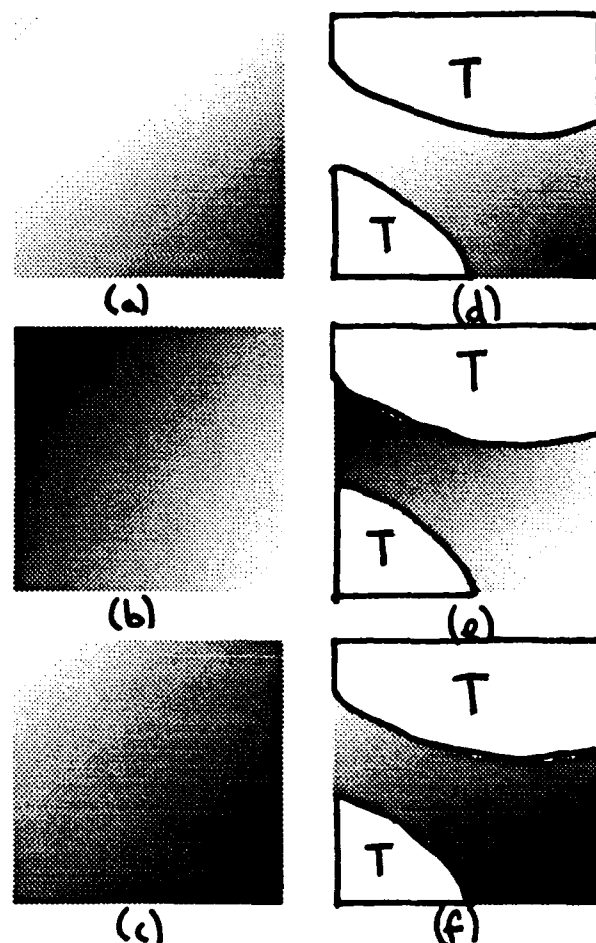


Figure 8: LTD vector components of an arbitrary rigid body motion (a) x-component (b) y-component (c) z-component (d) derived x-component (e) derived y-component (f) derived z-component

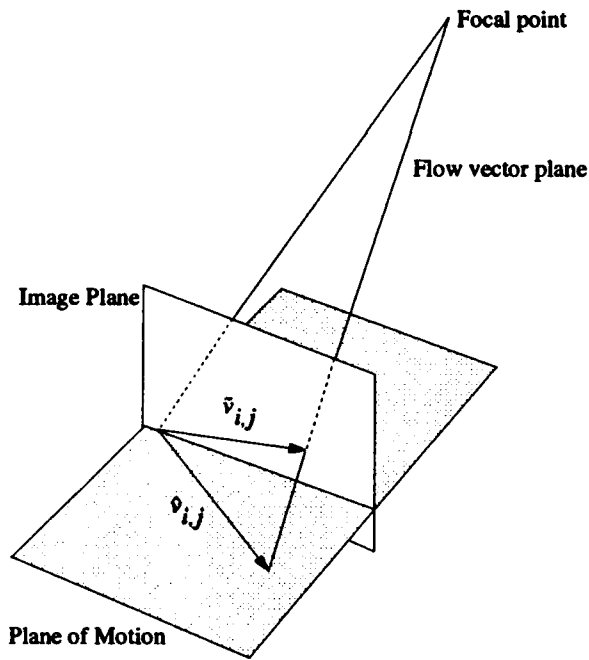


Figure 9: Motion constrained to a plane

plane of motion. We know from Section 2 that $v_{i,j}$ also lies in the plane determined by its corresponding flow vector $\tilde{v}_{i,j}$ and the focal point of the camera. The estimated direction of motion lies along the intersection of these planes. The estimated direction of motion $\hat{v}_{i,j}$ can be determined by intersecting these planes. Figure 9 shows the geometry, where the plane of motion is positioned so that it intersects the image plane at the base of the flow vector $\tilde{v}_{i,j}$. In terms of image geometry, this corresponds to intersecting the horizon line, determined by the plane of motion through the focal point, with a flow vector. The point of intersection is a Focus of Expansion for the local axis of translation (or a Focus of Contraction: which depends on the direction of the flow vector relative to the point of intersection). Computing the LTD in this case has been found to give extremely low errors (small fractions of a degree) in the estimated local translations.

Motion constrained to a plane is typical in terrestrial circumstances. Several indoor robotic environments involve robot motion constrained to a plane. In vehicular environments, the translational approximation is usually valid due to limitations in vehicle turning radii, meaning that the overall motion of a vehicle can be locally approximated as a translation.

2.2 Motion Constrained to an Undetermined Plane

Processing in the case of motion constrained to an undetermined plane is similar to that of motion constrained to a determined plane. The only difference is that an estimate of the plane of motion must first be recovered. Using the technique described in Section 2 the local trans-

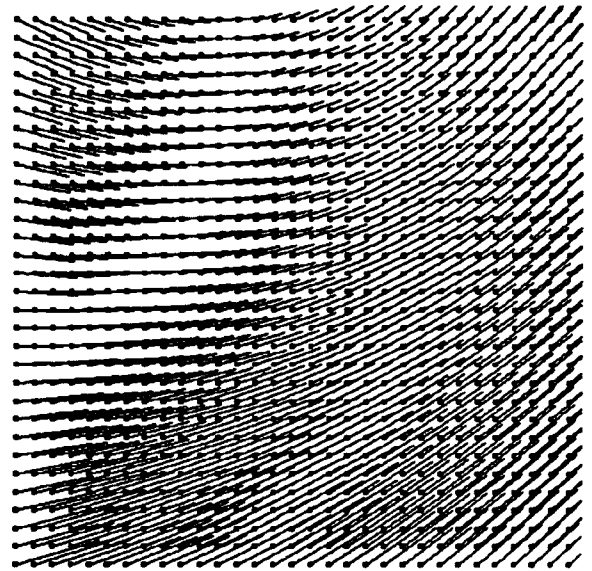


Figure 10: Optic flow field for a planar motion

lation is computed at each flow vector. Since the motion that produced these local translations is constrained to a plane, each of the local translations must be parallel to this plane. This constraint can be written as

$$\tilde{v}_{i,j} \cdot n = 0 \quad (5)$$

where n is a vector normal to the plane of motion. Using this equation, n can be computed by a linear least squares technique.

An example of processing in this case is shown in Figure 10 to Figure 12. Figure 10 shows the flow field produced by a rotation of 4.58° about the axis $(-1, 1, 2)$, followed by a translation of $(120, 20, 50)$. Units are given in pixels. This motion is constrained to lie in the plane perpendicular to the normal $(-1, 1, 2)$. However, the plane is unknown, so initially the local translation vectors must be computed by the method used for cases of arbitrary motion.

The angles between the correct local translational values and the derived local translational values shown are plotted in Figure 11. The error measure is shown in Figure 12. Since we have an error measure associated with each point describing the error of the translational approximation, we can select several positions of minimal error for use in Equation 5. Using the error measure from Equation 4 the best 15 local translations were selected for the least squares fit. The recovered plane normal is then $(-0.4107, 0.4129, 0.8129)$ which is off by an angle of 0.37° from the correct value. We can then use this estimate to evaluate the directions of motion using the technique for motion constrained to a determined plane from the previous section. The computed directions of motion are then shown in Figure 13 (d)-(f). Like the case of motion constrained to a known plane, there is very little error in the derived LTD vectors. The mean angle between derived and actual LTD vectors was 0.176° and the maximum angle was 1.274° .

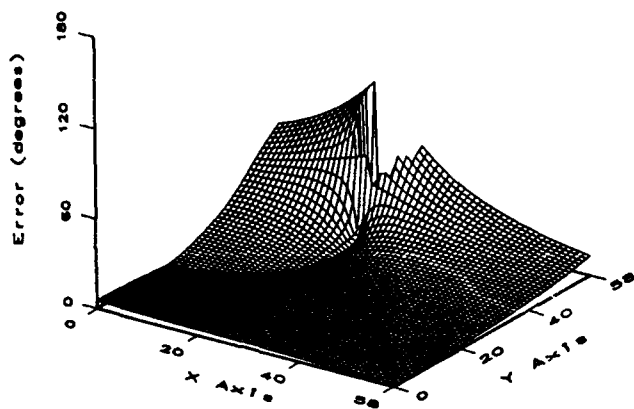


Figure 11: Actual errors for an unknown planar motion

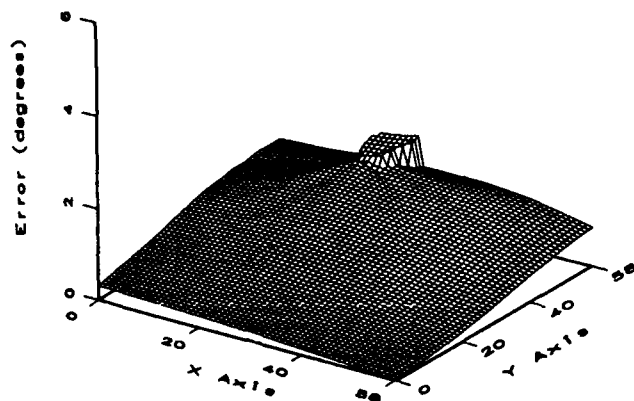


Figure 12: Evaluated error measure for unknown planar motion

2.3 Local Planarity and Rigidity-based LTD Estimation

Another algorithm for computing the LTD is based on the constraints provided by assuming motion relative to locally planar, rigid environmental surfaces. The algorithm begins by searching over the half-plane defined by a flow vector and the focal point of the camera as shown in Figure 1 (this plane is designated a half-plane because we only need to search over 180°). Each candidate LTD vector is used to solve for other LTD vectors in a local neighborhood by making an assumption of surface planarity within the neighborhood. The consistency of this local neighborhood of LTD vectors is then evaluated by calculating the relative depths of the LTD vectors. This results in an error measure which is associated with each candidate LTD vector. The candidate LTD vector with the lowest associated error is selected as the correct LTD vector. The remainder of this section describes this al-

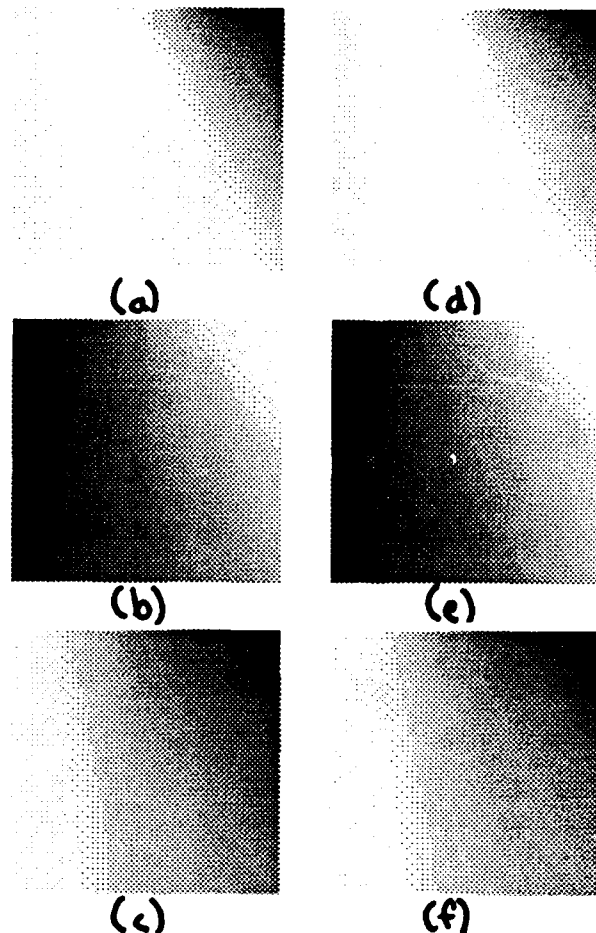


Figure 13: LTD vector components of an undetermined planar motion (LTD estimated using the determined planar motion technique) (a) x-component (b) y-component (c) z-component (d) derived x-component (e) derived y-component (f) derived z-component

gorithm in greater detail.

2.3.1 Local Planarity Assumption

Given a candidate LTD vector, we wish to solve for other nearby LTD vectors. In order to derive a relationship between LTD vectors within a neighborhood, we will assume that surfaces are locally planar. In this case directional derivatives of the LTD vectors along the image plane are constant. Let $\tilde{p}_{i-1,k}$, $\tilde{p}_{i,k}$, and $\tilde{p}_{i+1,k}$ be three collinear points on the image plane. Under the planar surface assumption, we have the following constraint

$$\frac{\hat{v}_{i+1,k} - \hat{v}_{i,k}}{\|\tilde{p}_{i+1,k} - \tilde{p}_{i,k}\|} = \frac{\hat{v}_{i,k} - \hat{v}_{i-1,k}}{\|\tilde{p}_{i,k} - \tilde{p}_{i-1,k}\|} \quad (6)$$

Letting $\hat{v}_{i,k}$ be the current candidate LTD vector, Equation 6 consists of two independent equations and six unknowns. The remaining equations needed to solve for these six unknowns can be provided by the LTD vectors' corresponding optic flow vectors. Figure 1 shows that the plane formed by a flow vector and the focal point

of the camera must include the LTD vector. This constraint can be written as

$$(\tilde{p}_{i-1,k} + \hat{v}_{i-1,k}) \times \tilde{p}_{i-1,k+1} = 0 \quad (7)$$

$$(\tilde{p}_{i+1,k} + \hat{v}_{i+1,k}) \times \tilde{p}_{i+1,k+1} = 0 \quad (8)$$

This provides four additional independent equations. Therefore, using the system defined by Equations 6, 7, and 8, we can solve for the neighborhood LTD vectors $\hat{v}_{i-1,k}$ and $\hat{v}_{i+1,k}$.

2.3.2 Error Measure

The final step in evaluating a candidate LTD vector is to construct an error measure from the neighborhood of derived LTD vectors. The relative depth of all the LTD vectors in a 3x3 neighborhood is calculated by positioning the candidate vector at the image plane. Using the depth values, a plane is fit to the neighborhood points. The error measure is defined as

$$\frac{1}{m} \sum_{i=1}^m (\alpha_i \tilde{p}_{i,k} - q_{i,k}) \quad (9)$$

where α_i is the depth scale factor and $q_{i,k}$ is the point of intersection of the fitted plane and the ray of projection defined by $\tilde{p}_{i,k}$. Section 3.1 shows how to solve for the depth scale factor α_i .

An example of processing an arbitrary motion using the rigidity-based method is shown in Figures 5 and 14. Figure 5 shows the flow field produced by a rotation of 5.73° about the axis (5, 4, 1), followed by a translation of (100, 25, -75). Units are given in pixels. Figure 14 (a)-(c) shows the correct local translational values as intensity plots of the vector components. Figure 14 (d)-(f) shows the local translational values that were derived from the optic flow field. Like the case of motion constrained to a known plane, there is very little error in the derived LTD vectors. The mean angle between derived and actual LTD vectors was 0.425° and the maximum angle was 2.647° .

3 Inferring Parameters of Motion from the LTD

In this section we develop a technique to recover the parameters of motion given a flow field and the LTD. The method presented in this section is based upon using rigidity to solve for the relative depth of environmental points associated with LTD vectors. The key result is that it is possible to infer the parameters of motion using only three determined LTD vectors computed from locations anywhere within the flow field. Thus, the inferring can be done with a sparse LTD field which may have been strongly filtered by the validity of the measures reflecting the translational fit. Once the relative depth has been determined, the solution for the parameters of motion becomes straightforward.

3.1 General Rigidity Constraint

In order to find the parameters of motion we will first solve for the relative depth of the LTD vectors using the rigidity constraint. Once the relative depth has been

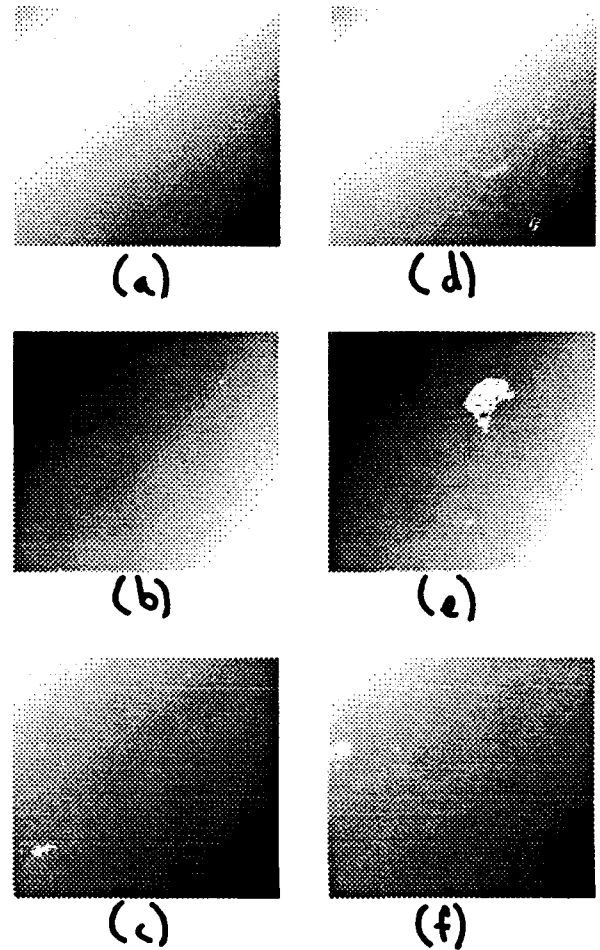


Figure 14: LTD vector components of an arbitrary rigid body motion (LTD vectors were derived using the local planar method) (a) x-component (b) y-component (c) z-component (d) derived x-component (e) derived y-component (f) derived z-component

determined, the solution for the parameters of motion becomes trivial.

Two LTD vectors $\hat{v}_{i,k}$ and $\hat{v}_{j,k}$ are assumed to have undergone identical rigid body motions. We wish to find the relative depth of these two vectors. Figure 15 shows the relationship between the two vectors. One of the vectors, $\hat{v}_{i,k}$, is fixed in depth so that it emanates from the image plane at the point $\tilde{p}_{i,k}$. The unknown depth of the other vector can be expressed as $\alpha \tilde{p}_{j,k}$ where α is some unknown scale factor. Since both of the LTD vectors are the result of the same rigid body motion, we have the following constraint

$$\|\alpha \tilde{p}_{j,k} - \tilde{p}_{i,k}\| = \|\alpha(\tilde{p}_{j,k} + \hat{v}_{j,k}) - (\tilde{p}_{i,k} + \hat{v}_{i,k})\| \quad (10)$$

Squaring both sides and solving for α , Equation 10 can be reduced to

$$\begin{aligned} & (2\tilde{p}_{j,k} \cdot \hat{v}_{j,k} + \hat{v}_{j,k} \cdot \hat{v}_{j,k})\alpha^2 - \\ & 2(\tilde{p}_{j,k} \cdot \hat{v}_{i,k} + \tilde{p}_{i,k} \cdot \hat{v}_{j,k} + \hat{v}_{i,k} \cdot \hat{v}_{j,k})\alpha + \\ & (2\tilde{p}_{i,k} \cdot \hat{v}_{i,k} + \hat{v}_{i,k} \cdot \hat{v}_{i,k}) = 0 \end{aligned} \quad (11)$$

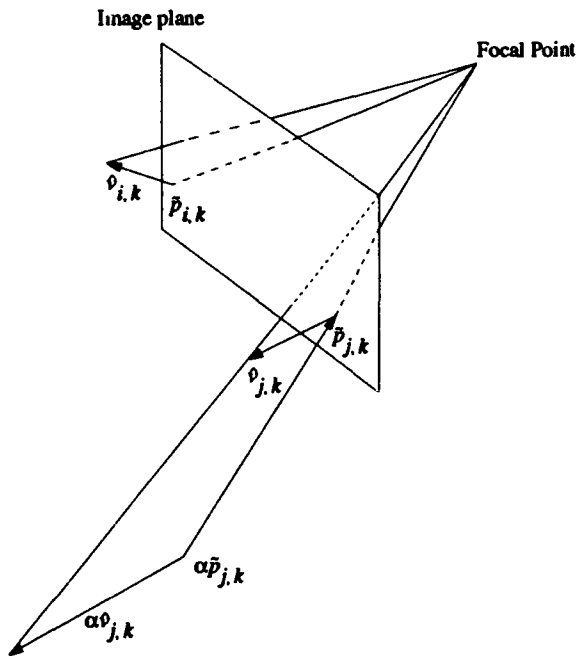


Figure 15: Relative depth of two LTD vectors

This equation is quadratic in α and results in two feasible solutions for the relative depth between two LTD vectors.

3.2 Inferring the Parameters of Motion

Once we have determined the relative depth between LTD vectors the estimation of the parameters of motion is trivial. The problem is equivalent to that of estimating the motion parameters from actual three-dimensional environmental surface positions. A rigid body motion can be expressed as

$$\alpha_{i,j} \hat{v}_{i,j} = r \times \alpha_{i,j} \tilde{p}_{i,j} + t \quad (12)$$

where r is the axis of rotation and t is the direction of translation. This expression is linear and can be solved using a least squares technique. The expression consists of six parameters and two independent equations. Therefore, it can be solved using a minimum of three (non-collinear) LTD vectors.

3.3 Motion Parameter Inference Results

The rigidity constraints were used to compute the parameters of motion from the derived LTDs presented in Section 2. The results are shown for the case of arbitrary motion, motion constrained to a determined plane, motion constrained to an undetermined plane, and the rigidity-based method applied to arbitrary motion. In the previous section we noted that the parameters of motion can actually be estimated using only three LTD vectors. The feasibility of estimating the parameters of motion from a minimal set of data is demonstrated in the results presented below.

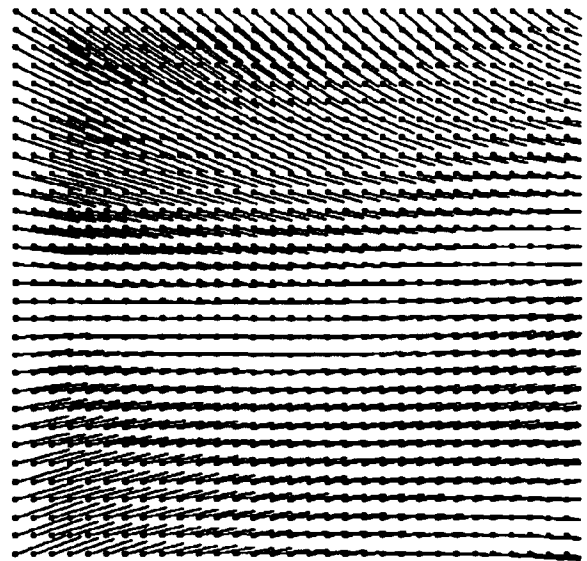


Figure 16: Optic flow field for motion relative to a curved surface

3.3.1 Motion Constrained to a Determined Plane

In the case of motion constrained to a determined plane, the LTD vector estimates tend to be highly accurate over an entire flow field. Typically, when using three LTD vectors selected at random from the derived local translations, the estimate of the axis of rotation and translation almost always are within a degree of the correct axes and the angle of rotation is determined to within a hundredth of a degree.

3.3.2 Motion Constrained to an Undetermined Plane

The case of motion constrained to an undetermined plane is similar to the case of motion constrained to a determined plane in that the LTD vector estimates are very good over the entire image. Three LTD vectors were selected at random from the derived local translations shown in Figure 13. The estimate of the axis of rotation was off by 0.99° , the angle of rotation was off by 0.04° , and the direction of translation was off by 0.83° .

3.3.3 Local Planar Method

The rigidity-based method presented in Section 3.1 is also capable of accurate LTD estimates over the entire flow field. Three LTD vectors were selected at random from the derived local translations shown in Figure 14. The estimate of the axis of rotation was off by 2.26° , the angle of rotation was off by 0.18° , and the direction of translation was off by 2.84° .

The camera was moved about a randomly curved surface. The optic flow field produced by this surface is shown in Figure 16. The three-dimensional environmental surface was reconstructed from this flow field. Figure 17 (a) shows a plot of the original surface. Figure 17 (b) shows the results of the surface reconstruction and Figure 17 (c) shows the resulting error in the

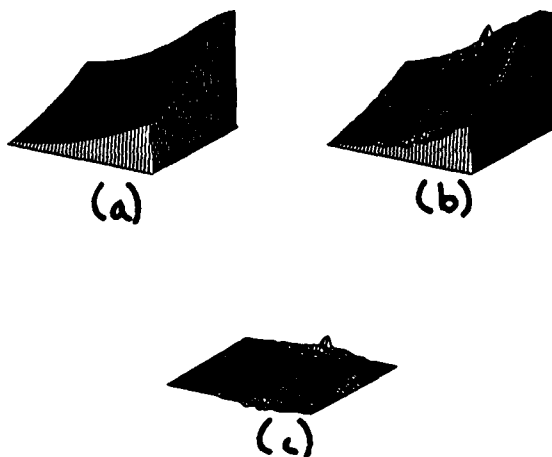


Figure 17: (a) Curved surface (b) Reconstructed surface (c) Error

reconstruction. The surface shown in this example is not planar. However, the reconstruction is fairly accurate, despite the violation of the planarity assumption. Experiments indicated that surfaces which are approximately planar in a local neighborhood can be successfully reconstructed. Therefore, any continuous surface can be reconstructed, given an appropriate density of optic flow vectors.

3.3.4 Arbitrary Motion

Using the error measure shown in Figure 7 and the derived LTD vectors shown in Figure 8, the three best LTD vectors were selected and used to compute the parameters of motion. The estimate of the axis of rotation was off by 8.13° , the angle of rotation was off by 1.09° , and the direction of translation was off by 12.02° . In the previous section it was shown that the minimum number of LTD vectors which can be used to estimate the parameters of motion is three. However, we can use a larger set of LTD vectors in a least squares procedure to obtain more accurate results. For example, when the ten best LTD vectors were used, the axis of rotation was off by 3.65° , the angle of rotation was off by 0.44° , and the direction of translation was off by 9.32° .

4 Summary and Future Work

We have introduced the local translational decomposition (LTD) as a low level representation of environmental motion which can simplify the inference of motion parameters from optic flow fields. We have found that this is particularly robust and simple for cases of motion constrained to a determined or undetermined plane, and motion relative to locally planar surfaces. In addition, it is possible to infer motion parameters from sparse LTDs.

Areas for further work include:

- Develop criteria to determine the the best set of estimated local translation vectors to estimate motion parameters in order to take advantage of the lim-

ited number of points for which the local translation needs to be determined to infer motion parameters.

- Investigate local translational analysis with the use of multiple cameras and longer image sequences.
- The local translation decomposition is similar to an array of localized looming detectors which determine whether things are coming towards or away from an observer at a particular image position. It may be possible to use such a distributed representation of motion relative to environmental surfaces to control navigation and other behaviors directly, without the inference of motion parameters from the LTD.
- The local translation approximation can be used as a criteria for computing flow to determine the LTD directly without the initial computation of a flow field. In the experiments presented above, we have assumed a uniformly dense flow field of high resolution. The translation procedure developed in [Lawton, 1982] was not applied to computed flow fields, but to successive images for which interesting points had been extracted from the initial image. Given distinctive features (at least two), it was possible to compute the direction of translation in a small image area. This use of the translational procedure can be seen as a local constraint on the determination of image displacements such that the corresponding environmental motion can be interpreted as being translational. For egomotion, this wouldn't require computation over the entire flow field since only three LTD vectors are needed. Where the translational approximation is poor there will be a large value in the error measure reflecting weaker confidence in the validity of the approximation.

References

- [Burger and Bhanu, 1989] W. Burger and Bir Bhanu. On computing a 'fuzzy' focus of expansion for autonomous navigation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 563-568, 1989.
- [Gibson, 1950] J.J. Gibson. *The perception of the visual world*. Houghton Mifflin, Boston, 1950.
- [Jain, 1983] R. Jain. Direct computation of the focus of expansion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:58-64, 1983.
- [Lawton, 1982] Daryl T. Lawton. Processing translational motion sequences. *Computer Vision, Graphics, and Image Processing*, 22:116-144, 1982.
- [Lee, 1980] D.N. Lee. The optic flow field: The foundation of vision. *Philosophical Transactions of the Royal Society of London, Series B*, 290:169-179, 1980.

Structure and Motion from Region Correspondences and Affine Invariants *

Chi-Yin Lee, David B. Cooper

Laboratory for Engineering Man/Machine Systems

Division of Engineering

Brown University

Providence, RI 02912

Abstract

This paper proposes a region based method to solve the structure and motion problem. Region matching between images is done by using affine invariants. No feature extraction or segmentation is needed in the matching process. Having recovered the region matches, a closed form solution for the camera motion and the 3D structures in the regions can be obtained. Structure consists of location and orientation of local planar patch approximate to 3D surface.

1 Introduction

Two primary approaches to the estimation of 3D structure and camera motion are based on feature correspondence and optical flow, respectively. An interesting example of the first approach is that taken in [4], in which the object of interest is assumed to be a 3D planar surface patch that is roughly perpendicular to the optical axis of the camera (the shallow structure assumption). Then, the object is tracked and its depth is recovered by estimating four affine parameters through line matches. [3] is an example of the latter approach. They compute the first order optical flow and estimate the camera motion and the orientation of the planar regions. Our approach is a generalization where we assume the 3D surface patch is planar where the plane has arbitrary orientation and location, and estimate both this plane and the camera motion from two images. Both the process of estimating feature correspondence and of computing the optical flow are somewhat noise sensitive. Our approach is region based and less sensitive to the noise of individual features or data points.

The assumption in this paper is that objects of interest are well approximated by groups of planar patches, and each planar patch is small compared with the patch-to-camera distance. Then the weak perspective model applies for the projection of the 3D surface into image planes, and a pair of images of points on the 3D planar surface patch are related by an affine transformation [1, 4].

The primary contribution of this paper is a set of low computational cost explicit expressions for the location and orientation parameters for these 3D planar surface patches based on a pair of images and for the motion specifying the two camera positions from which the pair of images were taken. A second contribution is a low computation cost algorithm for

matching small corresponding regions in a pair of images through use of a new class of affine moment invariants. The region to be matched can be at any arbitrary position and all the data points inside the region are to be used. No feature extraction or segmentation is needed in the matching process.

2 Equations of the Apparent Motion

Consider the system shown in Fig.1, the camera is moving from frame 1 at time t_1 to Frame 2 at time t_2 through a rotation R followed by a translation T . Let G be a planar region on a 3D surface and P be a particular point on it. The 3D coordinates of P w.r.t. Frame 1 and Frame 2 are denoted by (x, y, z) and (x', y', z') , respectively. (u, v) and (u', v') are the image coordinates of P at t_1 and at t_2 .

By using the weak perspective projection, the projections (u, v) and (u', v') at t_1 and t_2 are as follows:

$$\begin{pmatrix} u \\ v \end{pmatrix} = f \begin{pmatrix} x \\ y \end{pmatrix} \quad \begin{pmatrix} u' \\ v' \end{pmatrix} = \frac{f}{k} \begin{pmatrix} x' \\ y' \end{pmatrix} \quad (2.1)$$

where f is the focal length of the camera and l, k are the depths (z component of the 3D coordinates) of the centroid of region G w.r.t. camera Frame 1 and camera Frame 2, respectively. With small rotation, the 3D coordinates of P w.r.t. camera Frame 1 and camera Frame 2 are related by:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 & -w_x & w_y \\ w_x & 1 & -w_z \\ -w_y & w_z & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \quad (2.2)$$

Represent region G by the equation $ax' + by' + z' + d = 0$ w.r.t. Frame 2. By combining equations (2.1) and (2.2), the 2D apparent motion of the projections of P between image 2 and image 1 can be expressed by:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{k}{f} \begin{pmatrix} 1 - aw_y & -w_x - bw_y \\ w_x + aw_z & 1 + bw_z \end{pmatrix} \begin{pmatrix} u' \\ v' \end{pmatrix} + \frac{k}{f} \begin{pmatrix} t_x - dw_y \\ t_y + dw_z \end{pmatrix} \quad (2.3)$$

Clearly, (2.3) represents an affine transformation for the apparent motion. Denote the six affine parameters as follows:

$$\begin{aligned} h_1 &= \frac{k}{f}(1 - aw_y) & h_4 &= \frac{k}{f}(1 + bw_z) \\ h_2 &= \frac{k}{f}(-w_x - bw_y) & h_5 &= \frac{k}{f}(t_x - dw_y) \\ h_3 &= \frac{k}{f}(w_x + aw_z) & h_6 &= \frac{k}{f}(t_y + dw_z) \end{aligned} \quad (2.4)$$

Thus, the projections of region G onto Image2 and Image1 are related by the affine parameters h_1, h_2, \dots, h_6 . Without loss of generality, we assume

*This work was partially supported by NSF Grant #IRI-8715774 and NSF-DARPA Grant #IRI-8905436

the camera focal length to be unity in the following derivations.

3 Problem Definition and Our Approach

In preceding section, we derived the expressions for the parameters of the affine transformation that describes the apparent motion of a planar region under weak perspective projection. In order to estimate the camera motion and the 3D scene structure, two problems have to be solved,

1. For each 3D planar surface patch, one in each image, called 2D region matching, and recover the associated transformation between images.
2. Based on results in (1), solve for the camera motion and 3D structure parameters of the regions

Before explaining the scheme to solve the first problem listed above, in next section, we first give explicit solutions for camera motion and 3D scene structure based on knowledge of the affine transformation describing apparent motion. Then, we come back to present the 2D region matching algorithm and the estimation of the affine transformation based on moment invariants.

4 Motion and Structure Estimation

4.1 Solving the Motion and the Structure Parameters

As shown in equation (2.4), a 2D matched region pair has six equations relating the affine parameters with the surface patch parameters and camera motion. The unknown parameters are $k, l, a, b, d, w_x, w_y, w_z, t_x, t_y, t_z$. However, not all of the unknown parameters are independent. Let (u'_A, v'_A) denote the projection of the centroid of G onto image 2. Then the centroid of G w.r.t camera frame 2 is (ku'_A, kv'_A, k) . By observing the last equation of (2.2), we obtain an equation relating the depths of the centroid of G w.r.t camera frame 1 and camera frame 2.

$$l = k(-w_y u'_A + w_x v'_A + 1) + t_z$$

Since (ku'_A, kv'_A, k) is the centroid of region G , it must also satisfy the plane equation $ax' + by' + z' + d = 0$. Thus, we have,

$$aku'_A + bkv'_A + k + d = 0$$

As a result, we have 8 equations for a given 2D matched region pair. In these 8 equations, it is observed that l is a scale factor which cannot be determined and $k, a, b, d, w_x, w_y, w_z, t_x, t_y, t_z$ are the 10 unknown parameters. Thus, it is impossible to solve for both the structure and the motion parameter given the apparent motion of a region that is the image of only one planar patch.

Now, suppose we know the apparent motion of two regions. In addition to the region G described before, we have another region G_B with equation $a'x' + b'y' + z' + d' = 0$ w.r.t. camera frame 2 and the depths of its centroid w.r.t. camera frame 2 and camera frame 1 are p and q respectively. Additionally, denote by (u'_B, v'_B) the projection of the centroid of G_B onto image 2 and thus the centroid of G_B w.r.t. camera frame 2 is (pu'_B, pv'_B, p) . Again, there are 8 equations for region G_B .

$$\begin{aligned} h_7 &= \frac{2}{q}(1 - a'w_y) & h_{10} &= \frac{2}{q}(1 + b'w_x) \\ h_8 &= \frac{2}{q}(-w_x - b'w_y) & h_{11} &= \frac{1}{q}(t_x - d'w_y) \\ h_9 &= \frac{2}{q}(w_x + a'w_z) & h_{12} &= \frac{1}{q}(t_y + d'w_z) \end{aligned}$$

$$q = p(-w_y u'_B + w_x v'_B + 1) + t_z$$

$$a'pu'_B + b'pv'_B + p + d' = 0 \quad (*)$$

Given two matched region pairs, 16 equations in 16 variables are obtained. Again, l is a scale factor which cannot be determined. As a result, we have an overdetermined system with 16 equations in 15 variables. Our approach to handle this overdetermined system is to use the first fifteen equations to solve for the unknown parameters. Then, the last equation (*) is used to verify the solutions. In [2] we obtained the following polynomial after some manipulations of the fifteen equations

$$\lambda_1 w_x^4 + \lambda_2 w_x^3 + \lambda_3 w_x^2 + \lambda_4 w_x + \lambda_5 = 0$$

where the coefficients λ_i are functions of $h_1, h_2, h_3, h_4, h_7, h_8, h_9, h_{10}$.

The above equation is a fourth degree polynomial in w_x . For a fourth degree polynomial, there is a closed form solution for the four roots. Given w_x , the other variables can be solved successively giving one set of solutions. Detailed derivation is in [2]. Thus, we have four solutions in total. Then equation (*) is used to eliminate redundant solutions. For each of the examples we tried, only one of the four solutions satisfied (*). A numerical example is presented in the next section to illustrate this.

4.2 Numerical Example to Illustrate the Choice among the Four Solutions

Consider the camera motion parameters $w_x=0.06, w_y=0.05, w_z=0.02, t_x=5, t_y=8, t_z=2$. Two planar patches are given, having $0.15x' + 0.05y' + z' - 153.75 = 0, 0.1333x' + 0.2y' + z' - 201.3333 = 0$, and centroids $(4,6,200)$ and $(20,15,150)$, respectively, in camera frame 2. Equation (2.4) is used to generate the affine parameters. The four solutions are obtained by solving the fifteen equations described before. In order to exploit equation (*) to verify the solutions, we compute the bias,

$$\text{where } \text{bias} = a'pu'_B + b'pv'_B + p + d'$$

The correct solutions should be the ones with zero bias. Table 4.1 shows the four solutions with their bias and the ground truth values.

4.3 Practical Considerations

In real applications, the recovered affine coefficients for the matched regions are corrupted by noise and thus the bias's for each of the four motion and structure solutions are not zero in general. One reasonable choice of solution is that having minimum bias. Since each solution specifies the camera motion and the parameters of the 3D planar surfaces associated with the two 2D regions, for each point inside the two regions in the reference image its corresponding location in the other image can be computed and the intensity difference between them can be computed. The sum of all the intensity differences associated with all the points in the two regions is called the *Projective intensity difference* Γ . Thus, an alternate way to select the solutions is to pick the one with minimum Projective intensity difference. We employed this method in our experiments.

5 Recovering the apparent motion

The property of affine apparent motion of a region between two images explained in section 2 enables us to use a powerful tool - *affine invariance*. Affine invariants are functions of geometric structure which remain unchanged under affine transformation. In the following paragraphs, we briefly describe the idea of 2^+D data, 2^+D affine invariants, algorithms to find 2D region correspondence between images using the invariants and to recover the associated affine transformation parameters.

5.1 2^+D Data

Regions g and g' , the projections into camera image planes 1 and 2, respectively, of the 3D planar region G , respectively, are related by an affine transformation. Let (u_i, v_i) , (u'_i, v'_i) be the i^{th} matched points pair in g and g' , respectively. Then

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} h_1 & h_2 \\ h_3 & h_4 \end{pmatrix} \begin{pmatrix} u'_i \\ v'_i \end{pmatrix} + \begin{pmatrix} h_5 \\ h_6 \end{pmatrix} \quad (5.1.1)$$

Let the centers of g and g' be (m_x, m_y) and (m'_x, m'_y) respectively. By simple calculation,

$$\begin{pmatrix} u_i - m_x \\ v_i - m_y \end{pmatrix} = \begin{pmatrix} h_1 & h_2 \\ h_3 & h_4 \end{pmatrix} \begin{pmatrix} u'_i - m'_x \\ v'_i - m'_y \end{pmatrix} \quad (5.1.2)$$

Assume G is a Lambertian surface. Then, (u_i, v_i) and (u'_i, v'_i) appear with the intensity, say I_i , in both frame 1 and frame 2. Multiplying both sides of (5.1.2) by I_i , we get

$$\begin{pmatrix} (u_i - m_x)I_i \\ (v_i - m_y)I_i \end{pmatrix} = \begin{pmatrix} h_1 & h_2 \\ h_3 & h_4 \end{pmatrix} \begin{pmatrix} (u'_i - m'_x)I_i \\ (v'_i - m'_y)I_i \end{pmatrix} \quad (5.1.3)$$

For convenience, denote the above equation as,

$$\begin{pmatrix} \alpha_{i1} \\ \alpha_{i2} \end{pmatrix} = \begin{pmatrix} h_1 & h_2 \\ h_3 & h_4 \end{pmatrix} \begin{pmatrix} \alpha'_{i1} \\ \alpha'_{i2} \end{pmatrix} \quad (5.1.4)$$

From (5.1.4), it is realized that $(\alpha_{i1}, \alpha_{i2}), (\alpha'_{i1}, \alpha'_{i2})$ are still an affine pair. Thus, we construct two new data sets, $\{(\alpha_{i1}, \alpha_{i2})\}$ for all points in g and $\{(\alpha'_{i1}, \alpha'_{i2})\}$ for all points in g' . These data sets are related by parameters h_1, h_2, h_3, h_4 and contain information about not only the location of the image points but also about their intensities. The more interesting thing is that even with the additional intensity information, the dimension of the data set remains two and not three. So, we call them 2^+D data sets.

5.2 2^+D Affine Moment Invariants

In [5], a new framework is introduced for generating affine moment invariants. The particular moment invariants developed are the eigen-values of certain matrices, whose coefficients are algebraic functions of the location of 2D data. The computational cost of computing these invariants is low. As mentioned in the previous section, the dimension of the 2^+D data is two. Thus, we can apply the 2D moment invariants directly to our 2^+D data and we called them 2^+D moment invariants. In our experiment, five moment invariants are used, which are the eigen-values of a 2×2 and a 3×3 matrix. Thus, little computation is involved in the evaluation of the invariants.

5.3 Region Matching and Recovery of the Affine Parameters

In [2], we introduced a two stages scheme to perform region matching and to recover the associated affine transformation using 2^+D affine invariants. The idea of matching is to first compute the affine invariants for the region to be matched in the reference image, then locate the matched region in the second image such that its invariants are closest to that of the reference region. Having found the matched regions pair, the invariants also permit a trivial computation that provides a first estimate of the affine transformation. We then get an improved estimation of this affine transformation. Details of the algorithm are given in [2].

6 Experiments

6.1 Experiment 1

This experiment simulates a general camera motion, described by the translation $T = (1.563, 1.172, 0.391)$ in focal units and rotation angles $\Omega = (0.05, -0.06, 0.08)$ in radians. The scene consists of two distinct planar surfaces (left and right), represented by equations $0.5x' + 0.2y' + z' - 19.531 = 0$ and $0.2x' + 0.6y' + z' - 23.438 = 0$. Fig.6.1(a) and Fig.6.1(b) show the images taken before and after the camera motion, respectively.

Two circular regions are chosen in the reference image as shown in Fig.6.1(b). The matching results are depicted in Fig.6.1(a). In Table 6.1, we present the results of the recovered motion and structure parameters along with the ground truth values.

We see that the recovered values of the camera motion and the structure parameters are in good agreement with the ground truth values. Take the left plane as an example: the true normal to the plane and the recovered one differ only by 4.5 degrees and the error in the depth is 0.2%.

6.2 Experiment 2

This experiment is based on two images of a real scene taken by a moving camera. Fig.6.2(a) and Fig.6.2(b) show the images taken by the camera before and after its motion. To begin, the algorithm partitions the reference image into 64 circular windows as shown in Fig.6.2(b). The windows are numbered from g_0 to g_{63} .

Prior to initiating the matching process, based on the intensity histograms, windows with small intensity variation are discarded because they do not contain sufficient information for reliable matching. Fig.6.2(c) and Fig.6.2(d) show the matching results.

It is observed that all the windows but g_{11} have good matches. One reason for the mismatch is that this window lies on two surfaces, thus resulting in a large affine intensity difference Δ . Such a large difference is easily detected by the system. Thus, the system collected all the regions which have good affine matches.

In section 4, we showed that two matched regions pairs are needed in order to recover the motion and structure parameters. So it paired g_{30} with each one of the other regions and computed the motion and structure for each pair in parallel. As mentioned in section 4.3, the projective intensity difference Γ is also computed which is a measure of the goodness

of the recovery. By using this measure, it is found that the recovery of the motion and structure parameters for all the pairs described above was good except for the parameters found by the matched regions pair of g_{30} and g_{20} . So the result found by that pair is discarded and the results of the others are listed in Table 6.2. In table 6.2, the mean and the standard deviation of the motion parameters and the structure parameter associated with g_{30} are also computed. The recoveries are highly consistent, especially in the estimations of w_x, t_x, d .

The 3D reconstruction of the regions is displayed in Fig.6.3 by setting l equal to 10000. The equation of the planar surface associated with g_{30} is formed by the means of a, b, d shown in Table 6.2 From the top left corner of Fig.6.3, we see that the recovered surfaces associated with g_{17}, g_{19}, g_{26} and the ones associated with g_{19} and g_{12} formed the two sides of the big box. The patches associated with g_9 and g_{10} are sitting on the top of the box. The surface of the book on the bottom left and the box on the far right are estimated well.

7 Conclusion

This paper presents a new approach to the estimation of 3D surface structure and camera motion, based on two images. 3D surface structure is approximated by planar patches. The solutions are explicit and reliability is possible because the pair of images can be taken from two positions far apart, i.e., using a large baseline and because the matching is area based and thus resistant to noise. The required matching that is used to solve for the parameters of the affine transformation describing apparent motion is computationally modest because we use geometric invariants. Our approach can be used for aligning in pairs aerial photos taken from completely different azimuth and elevation angles.

Acknowledgement

The authors would like to acknowledge Dr. Daniel Keren for useful discussions and suggesting the idea of refining the affine parameters.

References

- [1] D.B Cooper, Y.P. Hung, and G. Taubin. A new model-based stereo approach for 3d surface reconstruction using contours on the surface pattern. *Intl. Conf. on Computer Vision*, pages 78-83, 1988.
- [2] C.Y. Lee and D.B. Cooper. Structure from motion: A region based approach using affine transformation and moment invariants. *Technical Report, LEMS-111, Div. of Engineering, Brown University*, 1992.
- [3] S Negahdaripour and S. Lee. Motion recovery from images sequences using first order flow information. *IEEE Workshop on Visual Motion*, pages 132-139, 1991.
- [4] H. Sawhney and A. Hanson. Identification and 3d description of shallow environmental structure in a sequence of images. *IEEE Conf. on CVPR*, pages 179-185, 1991.

- [5] G. Taubin and D.B. Cooper. Object recognition based on moment invariants (or algebraic invariants). *Geometric Invariance in Computer Vision, Edited by Mundy and Zisserman, MIT Press*, 1992.

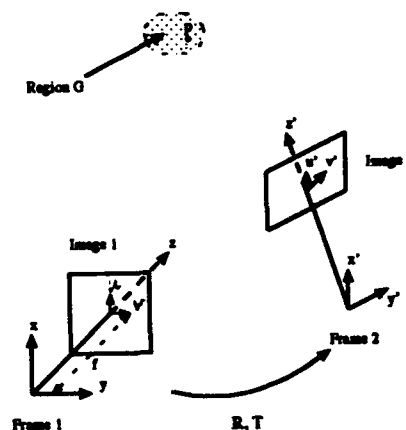


Figure 1

	Sol.1	Sol.2	Sol.3	Sol.4	true value
l	1.0	1.0	1.0	1.0	1.0
h	0.98931	0.982720	1.001187	1.001098	0.98931
q	0.751385	2.287143	1.643166	1.551075	0.751385
p	0.741987	2.430398	1.638967	1.534567	0.741987
a	0.133333	0.000000	-0.154775	-0.210727	0.133333
b	0.200000	-2.259781	0.000000	0.007207	0.200000
d	-0.997889	-0.916098	-0.998088	-0.997095	-0.997889
a'	0.150000	0.148948	-0.045779	-0.107571	0.150000
b'	0.050000	8.168292	0.398748	0.089732	0.050000
d'	-0.760536	-1.139293	-0.458436	-0.366400	-0.760536
w_x	0.060000	-0.004316	0.012769	0.012359	0.060000
w_y	0.050000	0.521550	-0.119175	-0.067117	0.050000
w_z	0.020000	0.028168	0.029644	0.030275	0.020000
t_x	0.024733	-0.403163	0.193574	0.181491	0.024733
t_y	0.039873	-0.027919	-0.007584	-0.007976	0.039873
t_z	0.009893	0.027776	-0.003957	-0.003213	0.009893
bias	0.000000	3.324593	1.238373	1.169728	

Notations are the same as in section 4.1

Table 4.1

	w_x	w_y	w_z	t_x/l	t_y/l	t_z/l
actual values	0.050	-0.050	0.050	0.075	0.050	0.019
recovered values	0.045	-0.045	0.045	0.069	0.070	0.017

	a	b	d/l	a'	b'	d'/l
actual values	0.150	0.200	-0.998	0.200	0.050	-1.198
recovered values	0.128	0.138	-0.998	0.084	0.313	-1.184

Notations are the same as in section 4.1

Table 6.1



Figure 6.1a



Figure 6.2a

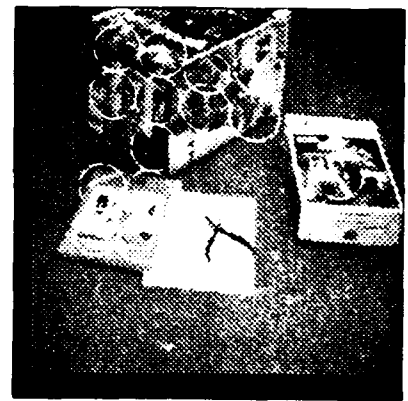


Figure 6.2c



Figure 6.1b

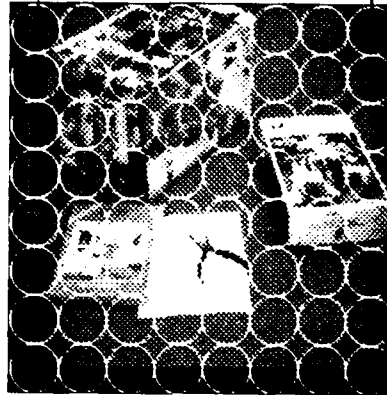


Figure 6.2b

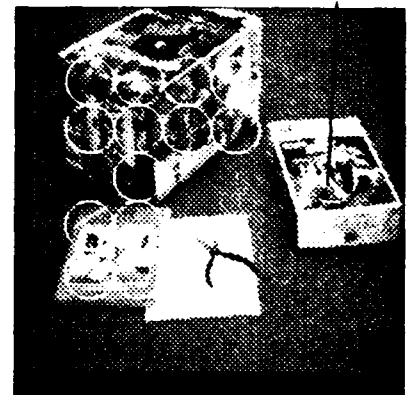


Figure 6.2d

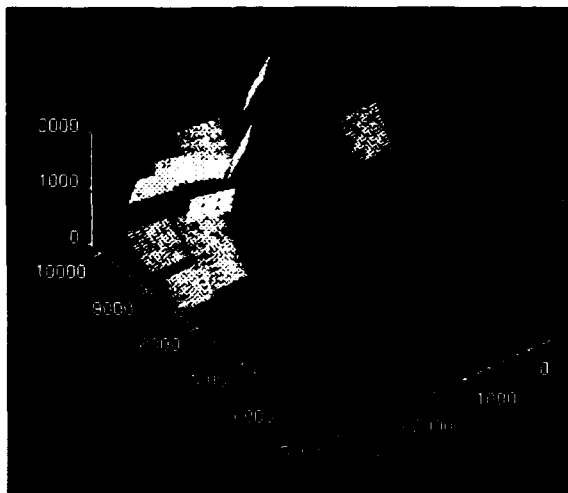


Figure 6.3

q'_1 (recovered by pair q_{30} and q'_1)	w_x	w_y	w_z	t_x/l	t_y/l	t_z/l
9	-0.064	-0.037	0.118	0.053	-0.051	0.028
10	-0.071	-0.053	0.125	0.069	-0.059	0.023
12	-0.063	-0.043	0.121	0.060	-0.051	0.023
17	-0.121	-0.112	0.129	0.129	-0.110	0.020
18	-0.053	-0.030	0.118	0.047	-0.040	0.027
19	-0.059	-0.033	0.118	0.049	-0.046	0.029
26	-0.090	-0.056	0.118	0.072	-0.084	0.031
33	-0.062	-0.035	0.118	0.052	-0.049	0.028
34	-0.074	-0.042	0.118	0.059	-0.061	0.029
41	-0.087	-0.050	0.118	0.066	-0.073	0.030
42	-0.115	-0.060	0.118	0.082	-0.100	0.033
mean	-0.078	-0.050	0.119	0.067	-0.065	0.027
standard deviation	0.021	0.021	0.003	0.022	0.022	0.003

q'_1 recovered by pair q_{30} and q'_1	a	b	d/l	a	b	d/l
9	-1.050	0.000	-0.955	-0.114	0.239	-0.899
10	-0.833	0.075	-0.972	-0.429	0.153	-0.909
12	-1.008	0.076	-0.968	0.2987	-0.676	-1.098
17	-0.454	0.076	-0.990	0.250	0.185	-0.981
18	-1.275	0.103	-0.950	0.365	0.565	-0.936
19	-1.143	0.000	-0.951	0.252	-0.724	-1.077
26	-0.686	-0.012	-0.962	0.186	0.258	-0.940
33	-1.089	0.000	-0.954	-0.771	0.152	-0.987
34	-0.911	0.000	-0.958	-0.694	0.154	-0.982
41	-0.777	0.000	-0.960	-0.574	0.016	-0.987
42	-0.587	0.000	-0.964	-0.521	0.042	-0.990
mean	-0.892	0.022	-0.962			
standard deviation	0.23	0.039	0.010			

Notations are the same as in section 4.1

Table 6.2

Section XIII

Object Recognition Using Invariance/Constraints

INVARIANT OBJECT RECOGNITION: A Model Evolution Approach

Peter W. Pachowicz

Center for Artificial Intelligence and Department of Systems Engineering
George Mason University, Fairfax, VA 22030
email: ppach@aic.gmu.edu

Abstract

A learning-based approach to object recognition under variable object characteristics is presented. The approach supports system capability of recognizing objects in dynamic environments by adapting the object models to perceived changes in object characteristics --- for example, caused by variable perceptual conditions. This adaptation is performed by the evolution of object models over attribute space, which is realized by integrating within a close loop a vision module with an incremental learning module. While the initial acquisition of object models is driven by a teacher, the later evolution of these models is performed over a sequence of images without the help of a teacher. Object models are applied to recognize objects on the next images. The effectiveness of such recognition and object extraction is monitored, and when it is decreasing the system selects new training data and activates learning processes to improve its models. These processes are related to active modeling performed by a system through the interaction with the dynamic environment. We have implemented the model evolution approach within a system and tested it for gradually changing resolution and lighting conditions. The experiments presented have

This research was conducted in the Center for Artificial Intelligence at George Mason University. The Center's research is supported in part by the National Science Foundation under grant No. IRI-9020266, in part by the Defense Advanced Research Projects Agency under the grant No. N00014-91-J-1854, administered by the Office of Naval Research, and the grant No. F49620-92-J-0549, administered by the Air Force Office of Scientific Research, and in part by the Office of Naval Research under grant No. N00014-91-J-1351.

been performed for recognition and segmentation of texture areas on a sequence of gray scale images.

1. Introduction

Most research on object recognition has been focused on learning and recognizing objects under stationary perceptual conditions such as lighting, resolution and positioning. Relatively little has been done on the problem of recognizing objects under dynamic conditions, particularly when the change of these conditions influence the change of object characteristics. This problem is particularly severe for object recognition in outdoor environments where the variability of perceptual conditions is extremely large [Bhanu et al., 1989, 1990].

To avoid some problems with the variability of object characteristics under the change of perceptual conditions, we can apply, for example, (1) domain specific feature selection, (2) active vision, or (3) model projection and adjustment. All of them, however, have significant limitations. Feature selection [Tsatsanis and Giannakis, 1992] bases on the idea of selecting/building such features that are sensitive to a given object in a wider range of perceptual conditions. But in practice, these features cause larger misclassification error. Active vision [Bajcsy, 1988] approach bases on the idea of manipulating camera parameters to maintain the same perceptual conditions. The problem with the variability in resolution, for example, can be mitigated through camera adjustment and the application of multiscale operators. But with the increase in the number of independently moving objects the vision system is becoming overloaded by camera manipulations. The problem remains with the other conditions. Finally, model prediction and adjustment bases on the projection of perceptual conditions that can occur in the future. Object models can be prepared respectively to these projections. For example, the problem with a pose

variability of structural objects can be practically eliminated by the generation of the aspect models [Ikeuchi, 1987]. But, these techniques seem not practical for other representations --- in particular, for texture recognition problem.

The above methods are applicable to the recognition of an object on a single image when relatively more time is given to perform the recognition task. But, they are not optimal when objects must be recognized, monitored and tracked through a sequence of images and the characteristics of these objects vary. These methods rely mostly on the *a-priori* provided physical, structural, functional and behavioral models of objects and the sensor. We agree that in the future the combination of them can be applicable to a very well defined and modeled problem. The possession of complete models, however, is questionable for most practical problems in machine perception and other complex large-scale systems.

Most approaches to object recognition do not adapt an object recognition system directly to the dynamic environment; i.e., they do not modify on-line object models to the changes of object characteristics. These methods use stationary models once acquired during the training phase, and they build a *transformation system* between these stationary models and the input data acquired from a dynamic environment. Therefore, we call such an adaptation an *indirect adaptation*. Such an approach requires that each condition influencing the change of object characteristics is represented in the *transformation system*. So, it suffers when the *transformation system* is not preprogrammed to deal with a specific perceptual condition which might not be known at the time of system development.

In the next sections, we present an alternative approach to the object recognition under variable perceptual conditions which directly adapts object models to perceived changes in object characteristics. This approach is outlined in Section 2. An example implementation of the developed method within the CHAMELEON '92 system is presented in Section 3. Finally, experimental results are presented in Sections 4.

2. Model Evolution Approach to Object Recognition

Model evolution approach to object recognition under variable perceptual conditions rely on the dynamic modification of object models according to the perceived change in object characteristics. It is done by close interaction of an integrated vision and learning system with the environment (learning from the environment). A vision system adapts to the changes in the environment by adapting the object models directly rather than building data

transformation modules fitted to the stationary models. This allows for capturing any variability of object characteristics without the knowledge about object properties and without building complex and dedicated modules serving the change of a given perceptual condition. Thus, an object model can be adapted to any combination of multiple perceptual conditions, the combination of which creates an infinite set of possible states. Moreover, the system can adapt to the change in the internal state of an object (e.g., to the change of the target heat signature --- in the Automatic Target Recognition domain).

Adaptation of object models to the perceived changes in perceptual conditions is particularly well suited to the problems where objects have been recognized once on an image, and they have to be recognized, monitored or tracked on the other images or over a sequence of images acquired under varying perceptual conditions. The application areas include, for example, scene annotation for navigation, autonomous surveillance, automated target recognition, industrial inspection, and material selection. This approach can also be superior to the understanding of an action performed by the object --- in the automatic damage assessment through the analysis of change in object state.

The model evolution proposed integrates a vision system and a learning system working within a close loop over a *set/sequence* of images (see Figure 1). The primary aspect of this approach is that a system has to recognize objects on images acquired over time. Images of such a sequence are affected by the variability of conditions under which objects are perceived. Object models once acquired through a dialog with a teacher are then applied to recognize objects on the next image. The *recognition effectiveness* of object models is continuously monitored and compared with the results on the previous image(s) or with stated minimum requirements. If this *recognition effectiveness* decreases, then learning processes are activated to improve the models' discriminating power. While the system learns initial object models from teacher-provided data (training examples), thereafter, the system has to update these models automatically without teacher help. It is done by automatic selection of new training data and the activation of the incremental learning processes.

Vision module selects new training data and activates the learning processes when needed. The modification of object models is performed by the learning module. The learning processes incorporate new training data in such a way that they modify the existing object models (rather than learning them from scratch) according to variability

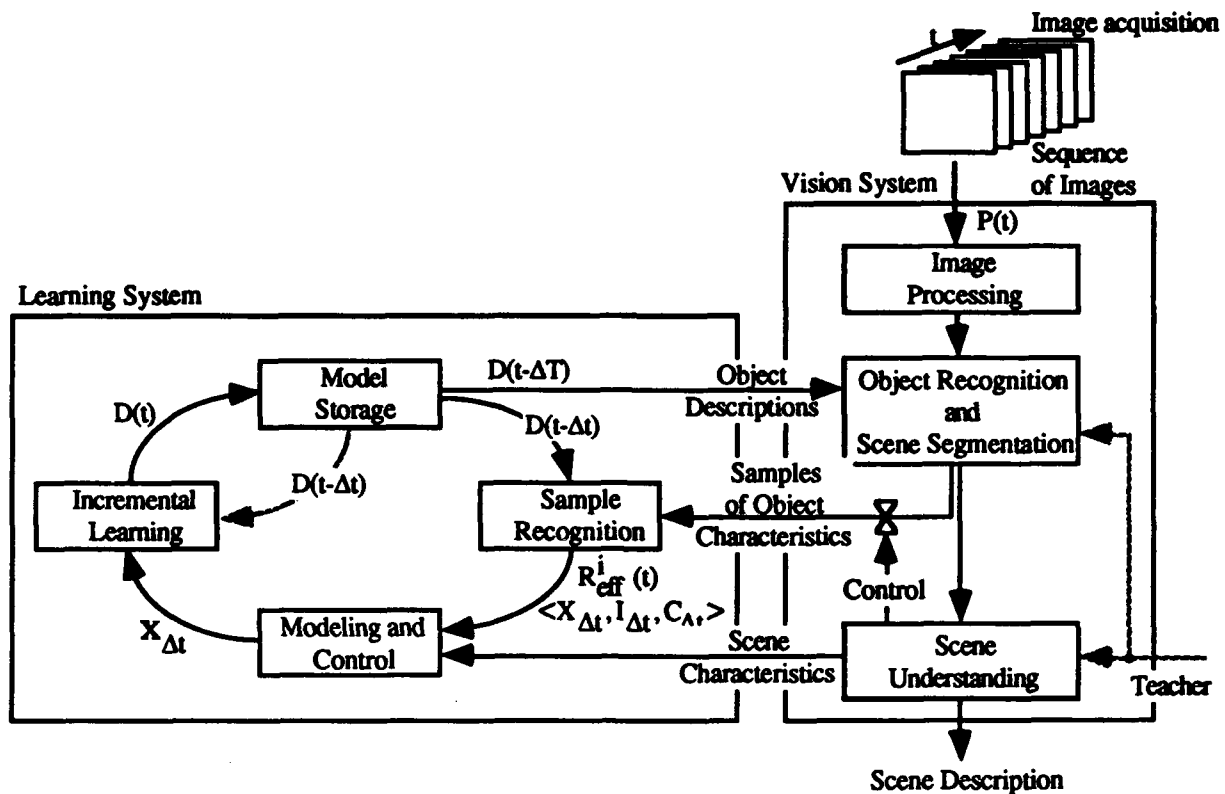


Fig.1 Architecture for model evolution integrating machine vision and learning systems; the CHAMELEON '92 system

of object characteristics represented by new training data. This process is performed by the incremental learning --- further generalizing object models.

Related research work has been reported by Goldfarb [1990] who introduced theoretical background to the model evolution integrating "Pattern Learning" of symbol formation and recognition with Artificial Intelligence of symbol manipulation. He suggested a Neural Net approach to system evolution but he has not provided an experimental confirmation of his approach yet. In the most recent work, Bobick and Bolles [1992] provide excellent motivation for the work on model evolution, focus on the evolution of representations for object recognition, and indicate stability problems in such systems. They learn object models everytime from a given image of a sequence. However, they do not evolve already existing models by new characteristic data which is the subject of our work. Our previous work include the definition of the learning-based approach to model evolution [Pachowicz, 1991], experiments with the CHAMELEON '91 semi-autonomous system of model evolution, and the analysis and improvement of the stability problems in the dynamic model evolution [Pachowicz, 1992]. The next section presents the CHAMELEON '92 fully autonomous model evolution system --- a system that evolves models without teacher help.

3. System Architecture and Implementation

The CHAMELEON '92 system, presented in Figure 1, has been created to investigate fully autonomous model evolution to the invariant object recognition in dynamic environments on the example of texture recognition. The domain of texture has been chosen because of high variability of texture attributional characteristics on the change in perceptual conditions (such as resolution, lighting, positioning, weather conditions).

3.1. Image data

The experimental input data was a sequence of six 256x256 black and white images (256 gray levels per pixel). The content of each image was simplified and each image was composed of six overlapping fabrics only. The images, presented in Figure 2, were affected by gradually changing resolution and illumination. The distance between the camera and the textured scene was gradually decreased to two thirds of the initial distance, and the light source was moved along with the camera.

3.2. Attribute extraction

In the first step, a single image is processed to extract texture features (attributes). For each pixel,

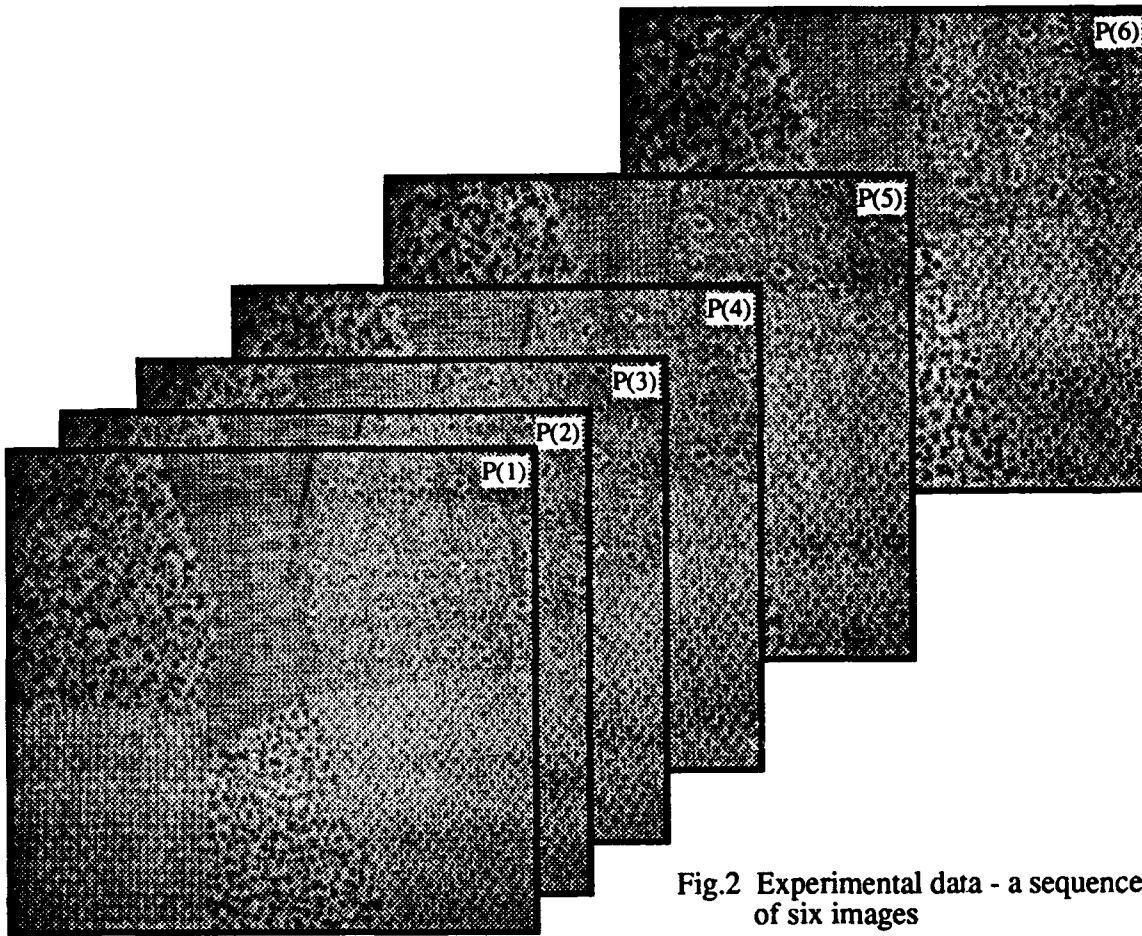


Fig.2 Experimental data - a sequence of six images

a vector of attributes is extracted characterizing its local neighborhood. We applied the modified Laws' [1980] method of texture energy measure [Hsiao and Sawchuk, 1989]. This three step procedure (1) convolutes an image with a given mask, (2) applies local averaging of the absolute responses, and (3) applies non-linear filtering to mitigate the borderline smoothing effect of previously applied averaging. The output is a vector of eight attributes corresponding to eight different convolution masks (i.e., S3S3, R5R5, E5L5, L5E5, E5S5, S5E5, L5S5, and S5L5). Numeric attributes were then quantized into subsymbolic intervals [Pachowicz, 1990]. We have chosen this method of attribute extraction because of the fast computation and quite well discriminating power. However, the other more powerful methods can be used as well.

Figure 3 presents an example distribution of texture attribute for a given class and over all six images of the sequence. First, analyzing attribute distributions we found that the distribution is multimodal for most attributes, classes, and images.

Second, the distribution of a given attribute vary significantly from one image to the other. The variability of perceptual conditions causes both the change of "shape" and the translation of the attribute distribution. These effects deteriorate model discriminating power when object models acquired from one image are applied to another image. And, this is why we have to adapt the vision system to a dynamic environment by evolving its object models.

3.3. Initial training phase

In the training phase, models of texture $D(t)$ are acquired from the first image of a sequence through the collaboration with a teacher. A teacher interactively indicates small sections of texture areas for the selection of the training data. Texture sections are then searched randomly to extract a given number of training data (examples) per class --- in our case, 300 examples (attribute vectors) per class. This data is forwarded to the learning module. The AQ14 learning program (for more details about this learning method and the program

see [Michalski, 1983]) is then applied to learn texture models (rule descriptions) from provided examples. However, the other learning programs can be applied to learn the models as well (e.g., the ID programs learning decision trees; [Quinlan, 1986]).

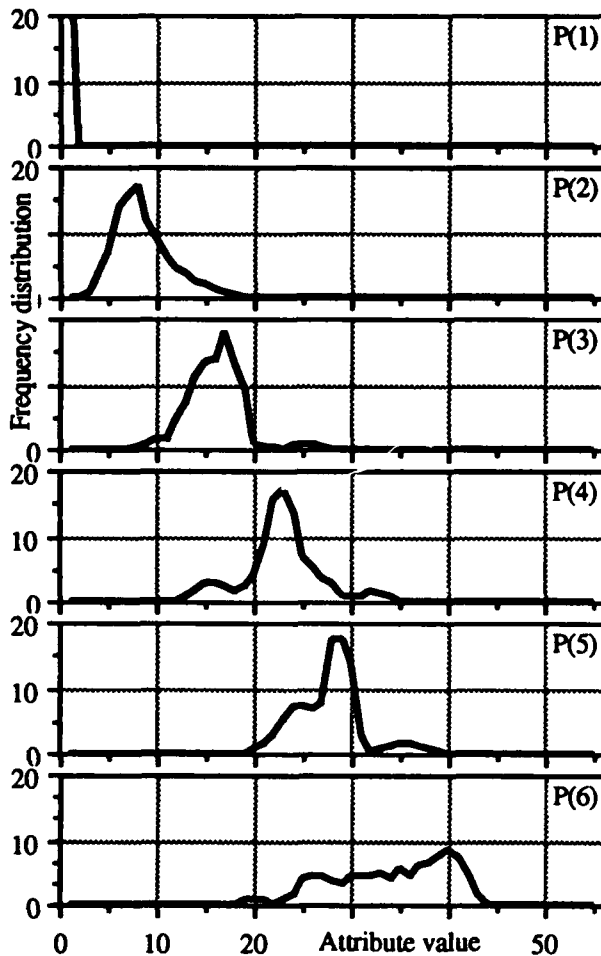


Fig.3 Example variability of attribute distribution over images of the sequence (for a given texture class)

Inductive learning acquires object models by drawing inductive inference from teacher- or environment-provided training examples. The learning process, incorporated by the AQ14 program, is performed for each class separately. Inductive learning applies a heuristic search through the attribute space incorporating inductive operators (generalization, transformation, correction, and refinement). Inductive learning is guided by background knowledge, which provides information about attributes, preference criteria, inference rules, heuristics, and program dependent procedures. The learning goal is to find the most preferred descriptions according to the preference criterion. The set of training examples of one class

is called a set of positive examples. With respect to this particular class, all other training data are negative examples. The AQ programs find object models over positive examples and no negative examples.

3.4. Recognition and segmentation

Once object models are acquired, they are applied to classify pixels and segment the image into class areas. Each attribute vector x from the incoming image is matched with the texture models. A single match of an attribute vector with a model of a given class produces a pair $[i - \text{classification decision}; c - \text{belief value}]$. The belief value c is maximum if the attribute vector is matched strictly (covered by) by the object model. In another case, the belief value is lower than the maximum value. The belief value is associated with the decision indicating the strength of the match. Since we have more than one object model, a vector of pairs $[i, c]$ is the output produced by the recognition algorithm applied to a single input attribute vector (at a given pixel position on an image) and for all object models (for $i=1$ to 6 ; the number of texture classes).

The belief values corresponding to the same classification decision are then locally averaged over the 3×3 window. This averaging, decision filtering, is repeated by a given number of iterations (i.e., 5 iterations). The final classification decision is made by yielding the decision of the highest averaged belief value. For better results, however, this process can be replaced by a more effective but computationally expensive relaxation method.

3.5. Evaluation of the model discriminating effectiveness

If an image is segmented, the evaluation process is run to determine the model discriminating power and to compare it with the previous results. This leads then towards a possible activation of the incremental modification of object models and the selection of new training data.

The evaluation is performed by automatically selecting some texture areas to compute recognition effectiveness measure; i.e., the average and the minimum recognition rates (or belief values) through all classes of texture. The texture areas are found by randomly searching for the uniform patches of 15×15 pixels of the same texture class through the entire image (see Figure 4a and 4b). (The 15×15 image window is considered by many researchers as the smallest window for the distinction of a texture.)

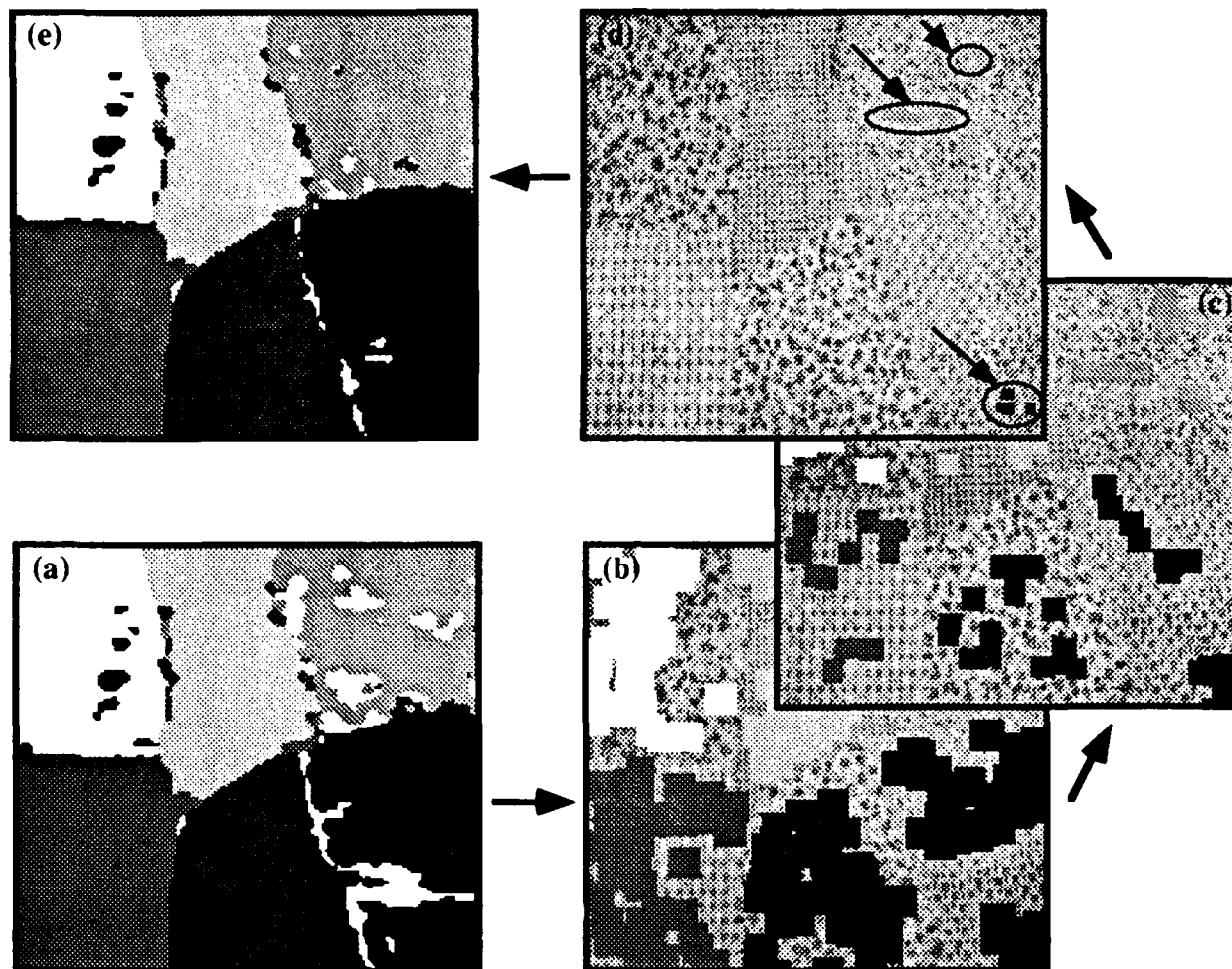


Fig.4 Extraction of new training data and its impact on the improvement of model discriminating power; a) segmented and annotated image by currently available models, b) randomly searched uniform patches, c) filtered intermediate areas carrying information about the change in texture characteristics, d) automatically selected new training examples (for the worst classes only), e) segmentation and annotation results for the updated texture models.

The patches of a single class are then divided into three groups: (group_area 1) typical texture areas (those that are recognized with the highest belief), (group_area 2) intermediate texture areas (those that represent change in texture characteristics), and (group_area 3) possible noisy texture areas (those that are mostly influenced by possible classification and segmentation errors). The division into these areas is done by computing the deviation from the average belief value for each texture patch. Only intermediate texture areas (group_area 2) are selected for the evaluation, which are shrunk twice to eliminate possible negative influence of the segmentation (see Figure 4c).

The recognition rate R_{eff}^i (see Figure 1) is then found for each i -th texture class by the analysis of recognition data (before they were smoothed) within the indicated group areas only. The

recognition rate for each class is calculated by the division of the number of correctly classified pixels to the total number of pixels covered by the group area. Then, the average recognition rate and the minimum recognition rate are calculated for the set of six texture classes.

3.6. Autonomous activation of incremental learning processes

The activation of incremental learning processes for model modification depends on the current evaluation of the model recognition results (model discriminating power) when compared with the results from the previous images. If these evaluation results deteriorate below a set or on-line adjusted threshold level, then the vision system activates the learning module. In the CHAMELEON '92 system applied to texture recognition, both threshold levels (for the average

and minimum recognition rates) have been set during the training phase run on the first image. These thresholds are constant through the remaining images of the sequence maintaining consistent *stop criteria* for the control of the evolution loop.

3.7. Selection of new training data

If the learning processes are activated then the areas used to evaluate model discriminating power are used to select new training data. For each class, the data selection process groups the area pixels into the following three files: pixels recognized by strict match (the maximum belief value), pixels recognized by flexible match (the closeness to the object model), and pixels not recognized correctly. New training data is then selected from the second and the third group. Only a limited number of pixels is extracted; i.e., the data is filtered to indicate the formation of new clusters only.

A set of new training data \mathcal{X} is then forwarded to the learning module. The number of new training examples depends on the performance of a given class. A very limited number of new training examples is allowed to be extracted (i.e., up to 20 examples per class). For the worse performing class, the most training examples are extracted. For the better performing class, practically no new data is extracted (see Figure 4d).

3.8. Incremental model modification

We incorporated an incremental learning to modify the once learned object models. The incremental learning methodology has already been implemented within several learning programs, i.e., within the AQ family of learning programs [Michalski and Larson, 1978], the ID family of learning programs [Utgoff, 1989], the INDUCE-4 program [Bentrup, et al., 1987], and conceptual clustering [Fisher, 1987, Gennari et al., 1989]. Incremental learning builds (modifies) object models dynamically according to newly provided evidence --- new training data. Therefore, this learning (model acquisition) technique was employed by us within the CHAMELEON '92 system to modify texture models. It also has been proved that incremental learning increases the speed of learning processes. Unfortunately, this learning technique can give slightly more complex models and somewhat worse recognition effectiveness.

In our system, newly extracted training examples along with object models are forwarded to the AQ14 incremental learning program. The results of the learning process are texture models modified according to the provided new object characteristics; i.e., the previous models are extended over the attribute space to include new training examples.

This modification of texture models is executed by their further generalization over the attribute space.

3.9. Verification of evolved models

Model evolution works in a closed loop, manipulating object models in order to adapt them to the changes in the object characteristics. This adaptation is performed in a two-loop system (see Figure 5). The external loop adapts models to a given image of a sequence, while the internal loop adapts models to the selected image data representing intermediate object characteristics. This schema was suggested by the early investigation of stability problems in the CHAMELEON '91 evolution systems [Pachowicz, 1992].

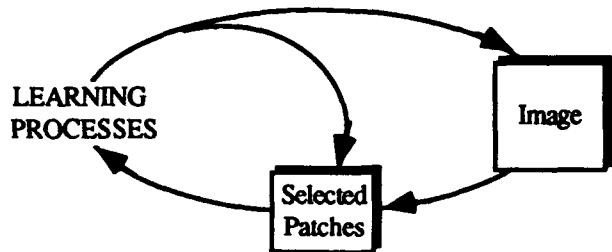


Fig.5 Two-loop evolution schema

The model evolution is competition oriented; i.e., a model of a given class is actually modified with the respect to the other classes. An extensive evolution of one class can cause weakness in the discriminating power of the other class. Therefore, a balance between the classes the system is trained to recognize should then be kept carefully. Progressing model evolution, the system has to verify the effect of the evolution. Regarding this progress the evolution can be repeated with the adjusted strategy and new training data. Particularly, the evolution must be continued as long as the recognition effectiveness is satisfied by *stop criteria*.

In the CHAMELEON '92 system, this verification is performed on the data characterizing the change in the object characteristics. Evolved models are applied to recognize those data. The recognition characteristics are computed, and then evaluated. If they not fulfill the assumed threshold levels for the average and minimum recognition rate, the evolution process is repeated but with new training data selected from the same areas. If they fulfill the assumed thresholds, this evolution loop is broken.

If the loop is broken, object models evolved over indicated image areas (Figure 4c) are later verified on the same image again. This verification repeats all processes of texture recognition, segmentation, and evaluation. If this evaluation does not satisfies

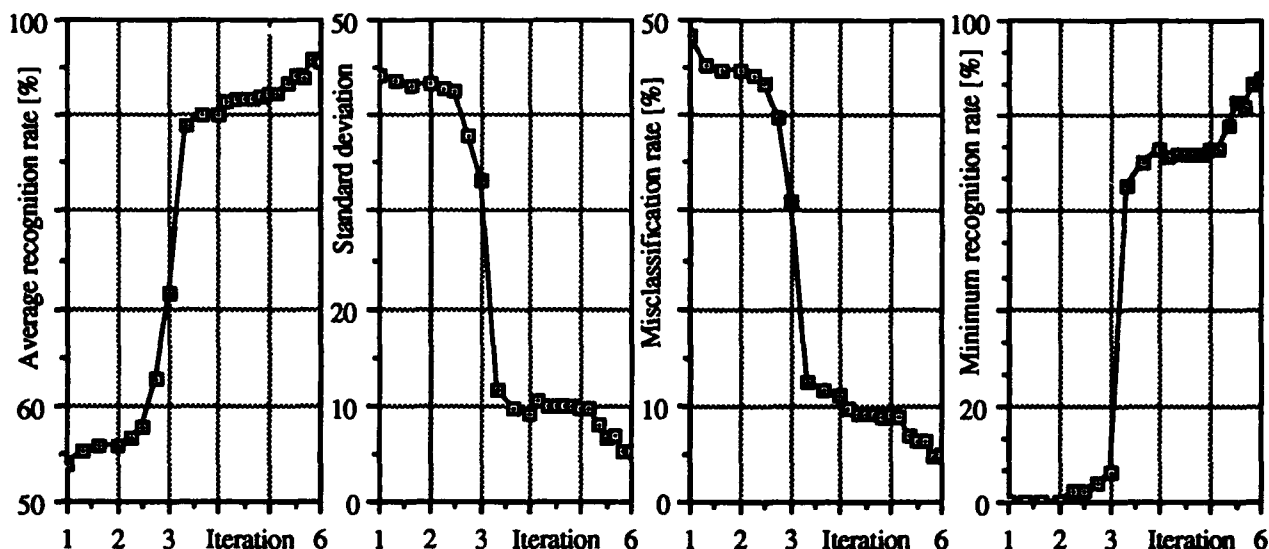


Fig. 6 Experimental results of system adaptation to the last (6th) image run via the model evolution through predecing sequence of images

the threshold conditions then the system activates model evolution over indicated image areas as described in previous sections. But, if this evaluation satisfies the threshold conditions then the system goes to the next image of a sequence. The evaluation of the evolution processes run over the example image presented in Figure 4a is illustrated on Figure 4e. The impact of the model modification by new training data shows the significant improvement in segmentation and annotation results for those two classes that were represented by new training data.

4. Experimental Results

4.1. Testing data and methodology

Considering an objective analysis of the system performance, evolution processes must be evaluated on different sets of data than data used in model evolution to modify object models. Therefore, testing data used to measure system performance was obtained from the same images but from widely spread image sections --- including sections close to the borderline between different texture areas. We indicated these sections interactively and extracted the testing data before the evolution processes were begun.

Since each image of a sequence is composed of six classes of texture, six testing datasets were obtained from each image. A single dataset contained 200 randomly selected (from indicated areas) testing examples characteristic for a single texture class. The testing phase was applied everytime when models were modified. The recognition

characteristics were obtained when evolved texture models were applied over and over again to the same image monitoring the evolution effect.

4.2. Recognition characteristics

Figure 6 shows the recognition characteristics for the last image of the sequence; i.e., when models were applied everytime to the sixth image. We monitor (i) average recognition rate over six texture classes, (ii) standard deviation from the average recognition rate, (iii) misclassification rate, and (iv) minimum recognition rate from the set of six classes. These diagrams are complemented by six images illustrating recognition and segmentation results on the sixth image over the consecutive iterations of model evolution (see Figure 7).

The experimental results show that initially learned texture models (i.e., from the first image) did not recognize some of the classes on the last image of the sequence. However, the model evolution over the next consecutive images has adapted texture models to changing texture characteristics. The system was able to improve its average recognition rate from 54% to 95%, while the minimum recognition rate was drastically improved from 0% to 89%. System maintained steady decrease in standard deviation of the recognition rates improving stability of the recognition system. In the same time, the misclassification rate was decreasing proving that texture models were not over-generalized; i.e., the competition with other class models kept *model boundaries* in balance.

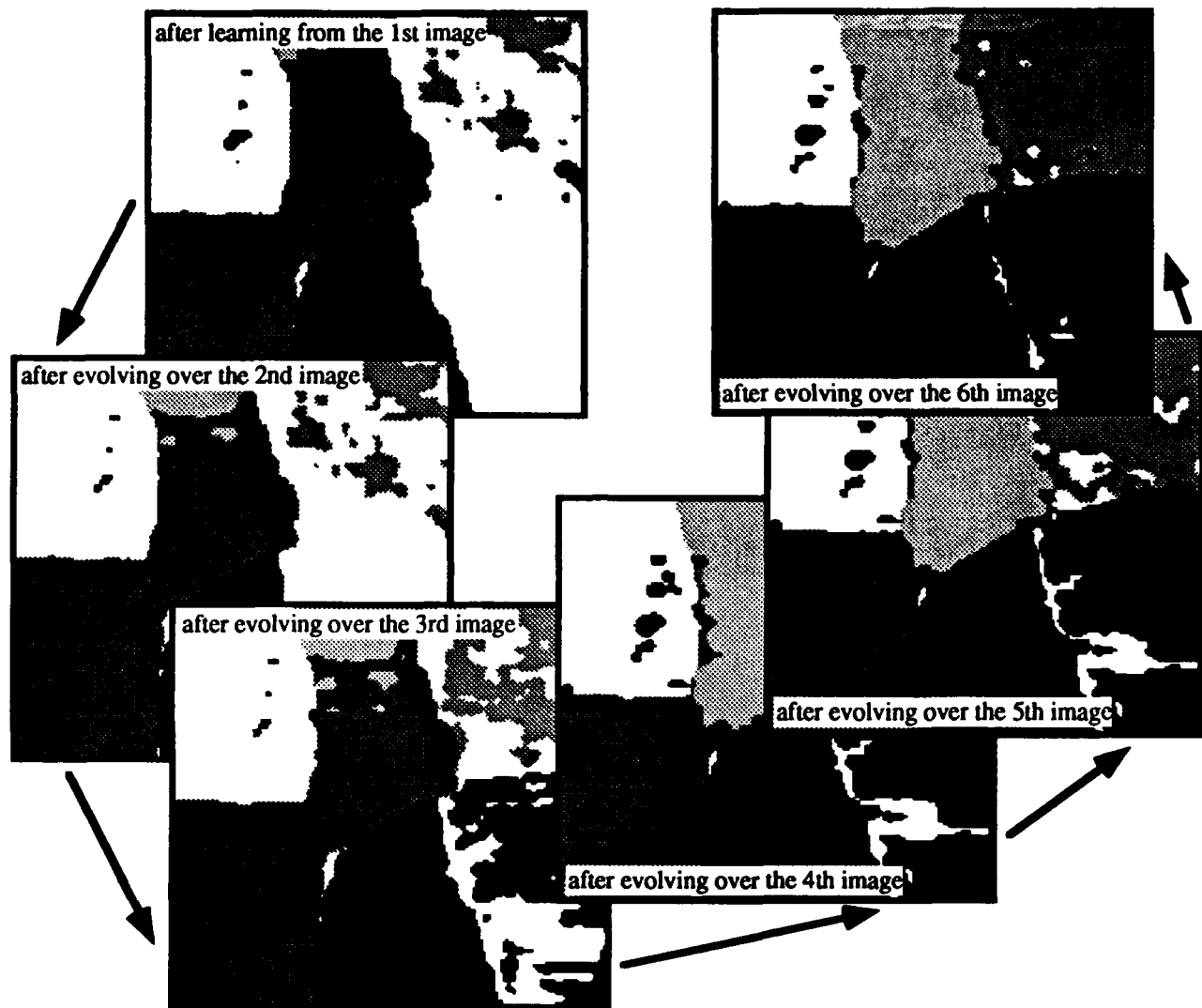


Fig.7 Model evolution results presented on the last image of the sequence from figure 2

5. Conclusions

The paper presented a method for object recognition under variable perceptual conditions that evolves object models adapting them directly to object new visual appearances. The evolution of object models is performed by integrating vision and incremental learning processes and maintaining system continuous interaction with the environment. The method has been implemented within the CHAMELEON '92 experimental system and tested successfully on texture recognition problem under changing resolution and lighting conditions. The system has been trained only on the first image. Thereafter, the system has worked autonomously on its own. The system was able to evolve its models over a sequence of images and to maintain its recognition capability.

References

- Bajcsy, R., "Active Perception", *Proc. IEEE*, pp.996-1005, 1988.
- Bentrup, J.A., G.J. Mehler and J.D. Riedesel, "INDUCE 4: A Program for Incrementally Learning Structural Descriptions from Examples", UIUCDCS-F-87-958, Computer Science Department, University of Illinois, Urbana, 1987.
- Bhanu, B., S. Lee and J. Ming, "Adaptive Image Segmentation Using a Genetic Algorithm", *Proc. DARPA Image Understanding Workshop*, Palo Alto, CA, pp.1043-1055, 1989.
- Bhanu, B., S. Lee and J. Ming, "Self-Optimizing Control System for Adaptive Image Segmentation",

Proc. DARPA Image Understanding Workshop, Pittsburgh, PA, pp.583-596, 1990.

Bobick, A.F. and R.C. Bolles, "The Representation Space Paradigm of Concurrent Evolving Object Descriptions", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.14, No.2, pp.146-156, 1992.

Fisher, D.H., "Knowledge Acquisition Via Incremental Conceptual Clustering", *Machine Learning*, Vol.2, pp.139-172, 1987.

Gennari, J.H., P. Langley and D. Fisher, "Models of Incremental Concept Formation", *Artificial Intelligence*, Vol.40, pp.11-61, 1989.

Goldfarb, L., "On the Foundations of Intelligent Processes - I. An evolving model for pattern learning", *Pattern Recognition*, Vol.23, No.6, pp.595-616, 1990.

Hsiao, J.Y. and A.A. Sawchuk, "Supervised Textured Image Segmentation Using Feature Smoothing and Probabilistic Relaxation Techniques", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.11, No.12, pp.1279-1292, 1989.

Ikeuchi, K., "Generating an Interpretation Tree from a CAD Model for 3D Object Recognition in Bin-Picking Tasks", *Int. J. of Computer Vision*, Vol.1, pp.145-165, 1987.

Laws, K.I., "Textured Image Segmentation", PhD Thesis, Dept. of Electrical Engineering, University of Southern California, Los Angeles, 1980.

Michalski, R.S. and J.B. Larson, "Selection of most representative training examples and incremental generation of VLI hypotheses: the underlining methodology and descriptions of programs ESEL and AQ11", Report 867, Department of Computer Science, University of Illinois, Urbana, 1978.

Michalski, R. S., "A Theory and Methodology of Inductive Learning", in *Machine Learning: An Artificial Intelligence Approach*, TIOGA Publishing, Palo Alto, CA, pp 83-134, 1983.

Pachowicz, P.W., "Integrating Low-Level Features Computation with Inductive Learning Techniques for Texture Recognition", *International Journal of Pattern Recognition and Artificial Intelligence*, Vol.4, No.2, pp.147-165, 1990.

Pachowicz, P.W., "Learning Invariant Texture Characteristics in Dynamic Environments: A model evolution approach", Report MLI-2-91, Center for

Artificial Intelligence, George Mason University, January 1991.

Pachowicz, P.W., "A Learning-Based Evolution of Concept Descriptions for an Adaptive Object Recognition", *Proc. IEEE Tools with AI*, Arlington, pp.316-323, 1992.

Quinlan, J. R., "Induction of Decision Trees", *Machine Learning*, Vol. 1, No. 1, 1986.

Tsatsanis, M.K. and G.B. Giannakis, "Object and Texture Classification Using Higher Order Statistics", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.14, No.7, pp.733-749, 1992.

Utgoff, P.E., "Incremental Induction of Decision Trees", *Machine Learning*, Vol.4, pp.161-186, 1989.

Quasi-invariant properties and 3-D shape recovery of non-straight, non-constant generalized cylinders

Mourad Zerroug and Ramakant Nevatia*
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, CA 90089-0273

Abstract

We address projective properties of the contours of Right generalized cylinders with a Planar, but not necessarily straight, axis and Circular, but possibly varying size, cross-sections (called Circular PRGCs). This class introduces many difficulties beyond the classes previously studied, such as straight homogeneous generalized cylinders (SHGCs) and their special case of surfaces of revolution (SORs), where the axis is straight, and planar right constant generalized cylinders (PRGCs), where the cross-section size is constant. Previous work on 2-D "ribbon" descriptions does relate to Circular PRGCs. However, it has not rigorously addressed or justified relationship between 2D descriptions and projection of 3D descriptions. In this work, we derive important rigorous *quasi-invariant* properties of Circular PRGCs and *invariant* properties for subclasses of Circular PRGCs. We show that the derived quasi-invariants are useful for 2D description of the projections of such primitives and for recovery of complete 3D object centered descriptions from the 2D contours. We demonstrate our claims on some examples.

1 Introduction and previous work

One of the major problems in computer vision is the recovery of shape of 3-D objects from a single 2-D contour image. This problem, known as *shape from contour*, is difficult because 2-D images contain appearances of real 3-D objects, which are dependent on, and hence may vary with, the viewpoint. In mathematical terms, the problem is under-constrained due to the loss of one dimension by the projective nature of the image formation. Human vision does show, however, that shape perception from such contours is largely invariant to changes in viewpoint. Previous work, ours

and others, has indicated that studying the projective properties of the contours, to find *invariant* properties, helps in scene perception in two important ways. First, they help detect the objects in the presence of noise, shadows and occlusion. Second, the properties provide important constraints for recovery of their 3-D shape. This work significantly extends the class of objects which can be so recovered. One important distinction from previous work is that for the new class, strict invariants are not found but *quasi-invariant* properties (defined formally later) are derived and proven to be equally effective.

Using contours as a source of constraints for shape has been the focus of research since the early days of computer vision. Early work addressed polyhedral objects using constraints on junction labelling [Clowes 1971] and face orientations [Kanade 1981, Mackworth 1973]. Subsequent efforts, such as [Gross & Boulton 1990, Malik 1987, Nalwa 1989, Nevatia & Binford 1977, Ponce et al. 1989, Sato & Binford 1992, Ulupinar & Nevatia 1990a, Ulupinar & Nevatia 1991, Zerroug & Nevatia 1993], have addressed curved surface objects. These objects introduce more difficulties as some of their contours, such as limbs and cusps (where the viewing direction is tangential to the surface), are inherently viewpoint dependent. To obtain rigorous properties, it is useful to study *classes* of objects. Of course, to be of interest, such classes should include *generic shape models* with the ability to generate a large set of everyday objects. Generalized cylinders (GCs) [Binford 1971], are one such adequate shape model. They have proved to be particularly suited for structured shape description of complex and articulated objects.

There is strong psychological evidence [Biederman 1987] that human perception of line drawings of complex objects is influenced by perception of arrangements of a small number of simple volumetric primitives. Those basic primitives, called *geons* (analogous to generalized cylinders), are characterized by different cross-section shapes, axis shape and sweeps. This indicates that it is sufficient to address shape recovery of

* This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-90-C-0078. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

a small set of primitive generalized cylinders for a computational approach to recovery of a large number of objects in our environment. A computational approach to the analysis and recovery of those primitives requires the study of their projective properties and their usage for recovering 3D shape.

A number of researchers have addressed the use of GCs and their projective invariant properties. Nalwa [Nalwa 1989] has proved that contours of surfaces of revolution (SORs), under orthographic projection, exhibit bilateral symmetry. Ponce *et al* [Ponce *et al.* 1989] have proved that in the (perspective) projection of straight homogeneous generalized cylinders (SHGCs), tangents to limbs at corresponding points intersect on a line which is the projection of the axis and exploited this property for detection of the projection of the axis of an SHGC from its image contours. Ulupinar and Nevatia [Ulupinar & Nevatia 1990a and b, Ulupinar & Nevatia 1991] have derived projective invariants of zero Gaussian curvature (ZGC) surfaces, SHGCs and planar, right, constant generalized cylinders (PRGCs). For instance, they have proved that cross-sections of SHGCs and limbs of PRGCs project onto "parallel symmetric" curves under orthographic projection. They have exploited those properties for recovering 3D shape from perfect contours. Sato and Binford [Sato & Binford 1992] and Zerroug and Nevatia [Zerroug & Nevatia 1993] derived and used projective invariant properties of SHGCs for solving the figure ground problem in real image contours.

The primitives addressed by previous work have either a *straight axis*, such as SORs and SHGCs, or a *constant cross-section size* such as PRGCs. Some natural objects, however, such as human and animal limbs and horns, have a combination of curved axes *and* varying cross-section size; some examples are shown in Figure 1. Departing from the previously addressed cases of straight axis *or* constant cross-section, to include objects with arbitrary 3D axes, cross-sections and sweeping functions, introduces many new difficulties. In this work, we address the class of GCs having *curved* (planar) axes with *non constant*, circular, cross-sections. Following the terminology of [Shafer & Kanade 1983], they can be called *circular planar right GCs* (Circular PRGCs) as the cross-section is orthogonal to the axis.



Figure 1 Sample Circular PRGCs.

Some previous efforts, such as [Brooks 1983, Nevatia & Binford 1977, Rao & Nevatia 1989], have used ribbons (2-D counterparts of GCs) as intuitive descrip-

tors of the *projection* of curved GCs, assuming that the 2-D descriptions correspond to the projection of the 3-D descriptions (ribbon axis and projection of 3-D axis, for example). However, the relationship between the 2-D descriptions and the projections of the 3-D descriptions was not rigorously addressed. It can be shown that in general they are not the same.

We were unable to find *invariant* properties of general Circular PRGCs (except for special cases), and we believe that none exist. The non-constancy of the cross-section and the curvature of the axis affect the contours in very complex ways. However, we have been successful in finding *quasi-invariant* properties (following the terminology of [Binford *et al.* 1987, Binford 1991]) that are useful for shape description and recovery. Quasi-invariance is a generalization of invariance. Invariant properties are properties with constant measure with respect to a set of parametric transformations (they hold independently of the parameters of the transformations¹). For example, the ratio of the lengths of two (3-D) parallel segments is known to be an orthographic invariant. Quasi-invariant properties are properties that may not be strictly constant, but their measure varies within a small range over a large set in the parameter space of the transformations. For example, the previous lengths ratio is a perspective quasi-invariant, as its value is within 10% of the actual one over 90% of the viewing sphere [Binford *et al.* 1987].

In this work, we derive important quasi-invariant projective properties of Circular PRGCs, invariant properties of their special cases, and show their application for shape description and recovery. Our analysis shows that a popular class of ribbons (so-called Brooks' ribbons, in the terminology of [Ponce 1988]) provides generally consistent descriptions (ones that correspond to projections of 3-D descriptions) of the projections of Circular PRGCs. Our recovery method need not be told that it is examining Circular PRGCs, rather it contains tests that can verify their presence. The recovery method assumes that the viewpoint is general and that both end cross-sections of viewed Circular PRGCs are visible.

We organize the discussion as follows. In section 2 we provide a mathematical analysis of the projected contours of Circular PRGCs. In section 3, we derive projective invariant properties for special Circular PRGCs and in section 4 we derive quasi-invariant properties for general Circular PRGCs. We also discuss relationship of the derived properties with previous work. In section 5, we discuss the application of the derived quasi-invariants for 2D description of Circular PRGCs and for rigorous recovery of complete 3D models of Circular

¹except perhaps on a set of measure zero in the parameter space; i.e. *almost everywhere*.

PRGCs from their 2D contours. We also demonstrate our methods on some examples. We conclude this paper in section 6.

2 Limbs and projections of Circular PRGCs

We begin by giving a formal definition of a GC, then proceed to the analysis of limbs and projections of Circular PRGCs. Throughout the analysis, orthographic projection will be assumed.

Definition1: a generalized cylinder (GC) is the surface obtained by sweeping a given cross-section curve C along a 3D (axis) curve A , while transforming it by a function r .

For GCs where the cross-section is orthogonal to the axis (Right GCs), the surface can be parameterized as follows (using a notation similar to [Ponce & Chelberg 1987]):

$$P(s, \theta) = A(s) + u(s, \theta) (\cos \theta \hat{n} + \sin \theta \hat{b}) \quad (1)$$

where $A(s)$ is an arclength parameterization of the axis curve A , $u(s, \theta)$ the cross-section function and \hat{n} and \hat{b} respectively the normal and binormal to the axis curve. For homogeneous cross-sections (i.e. rigid sweeps) $u(s, \theta) = r(s) \rho(\theta)$, where $r(s)$ is the scaling function and $\rho(\theta)$ the cross-section curve. Figure 2a illustrates this parametrization in the case of a Circular PRGC.

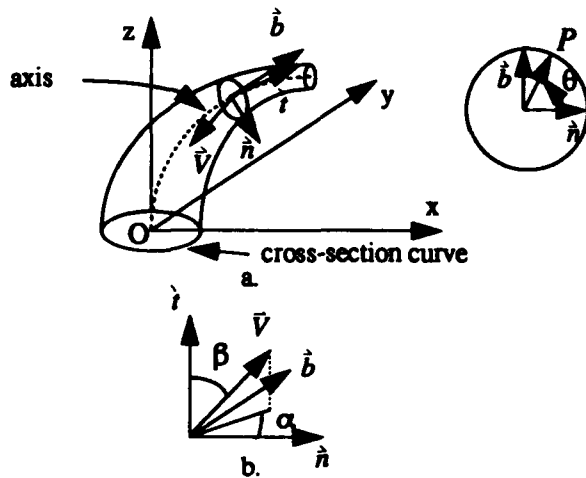


Figure 2 Generalized cylinder parameterization and viewing geometry

2.1 Deriving limbs of Circular PRGCs

For lack of space we omit the details of the limb derivation process as a detailed analysis is given in [Ponce & Chelberg 1987]. We limit the analysis to defining limb boundaries and expressing their equation. We further limit the analysis at this stage to Circular PRGCs (i.e. zero axis torsion and $\rho(\theta) = \rho$ and $\dot{\rho}(\theta) = 0$).

A point on the GC surface is a limb point *iff* the viewing direction is orthogonal to the surface normal at that point. The surface normal at a point $P(s, \theta)$ is given by

$$\hat{N}(s) = \frac{\partial P}{\partial s} \times \frac{\partial P}{\partial \theta} / \left\| \frac{\partial P}{\partial s} \times \frac{\partial P}{\partial \theta} \right\|, \text{ where } P = P(s, \theta) \quad (2)$$

Let $(\alpha(s), \beta(s))$ be the angular coordinates of the unit viewing vector \hat{V} in the Frenet-Serret frame $(\hat{i}(s), \hat{n}(s), \hat{b}(s))$ (Figure 2b), then

$$\hat{V} = \cos \beta(s) \hat{i}(s) + \sin \beta(s) \cos \alpha(s) \hat{n}(s) + \sin \beta(s) \sin \alpha(s) \hat{b}(s) \quad (3)$$

Using the Frenet-Serret theorem [Millman & Parker 1977] relating the above basis vectors and their derivatives and writing the orthogonality of $\hat{N}(s)$ and \hat{V} yields the following limb equation:

$$\sin \beta(s) [1 - \kappa(s) \rho r(s) \cos \theta] \cos(\theta - \alpha(s)) - \rho \dot{r}(s) \cos \beta(s) = 0$$

where $\kappa(s)$ is the curvature of the axis and $\dot{r}(s)$ the derivative of $r(s)$. Assuming $\sin \beta(s) \neq 0$ ², this equation can be rewritten as:

$$[1 - \kappa(s) \rho r(s) \cos \theta] \cos(\theta - \alpha(s)) = \rho \dot{r}(s) \cot \beta(s) \quad (4)$$

The limb equation usually has two solutions $\theta_i(s)$ for $i = 1, 2$. We can derive limb equations for special cases such as SORs and Circular PRGCs (constant size cross-section; note the difference with the more general Circular PRGCs where the cross-section is non constant). For SORs, the axis is straight ($\kappa(s) = 0$) and we obtain the SOR limb equation:

$$\cos(\theta - \alpha(s)) = \rho \dot{r}(s) \cot \beta(s) \quad (5)$$

which, for every s yields two solutions of opposite angular distance from $\alpha(s)$ (Figure 3.a).

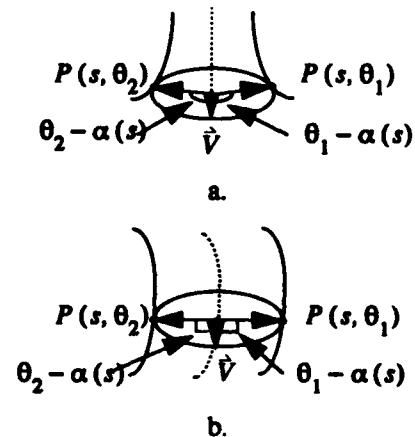


Figure 3 Limb points properties. a. SOR b circular PRGC

For Circular PRGCs, the sweeping function is constant ($\dot{r}(s) = 0$) and we obtain the equation

²this does not hold only when the viewing direction \hat{V} is parallel to \hat{i} , a non general viewpoint for which the limb equation has an infinite or zero number of solutions.

$(1 - \kappa(s)pr(s)\cos\theta)\cos(\theta - \alpha(s)) = 0$, which implies

$$\cos(\theta - \alpha(s)) = 0 \quad (6)$$

since $\sin\beta(s) \neq 0$ and $1 - \kappa(s)pr(s)\cos\theta = 0$ would imply $1/\kappa(s) = R(s) < pr(s)$; i.e. the object self intersects, an undesirable irregularity ($R(s)$ is the radius of curvature of the axis and $pr(s)$ the cross-section radius). Thus, in the case of Circular PRCGs, limb points are determined by $\theta = \pm\pi/2 + \alpha(s)$; i.e. two diametrically opposite points (Figure 3.b).

For the general case of Circular PRCGs ($\kappa(s) \neq 0$ and $\dot{r}(s) \neq 0$), the relationship between the two limb points is not as straightforward as the two sub-cases discussed above. However, in section 4, we will show that they *do* have a well behaved relationship.

2.2 Deriving projections of Circular PRCGs

The orthographic projection of the (3D) Frenet-Serret frame ($\hat{t}(s), \hat{n}(s), \hat{b}(s)$) on a plane orthogonal to \hat{V} gives a 'moving' local 2D frame in the image for each value of s . The relationship between the 3D and the 2D frames is as follows (we will omit the argument s):

Let $P = t\hat{t} + n\hat{n} + b\hat{b}$ be a point expressed in the 3D frame, then its projection, p , on a plane orthogonal to \hat{V} is given by

$$p = (-\sin\beta t + \cos\beta\cos\alpha n + \cos\beta\sin\alpha b)\hat{u} + (-\sin\alpha n + \cos\alpha b)\hat{v} \quad (7)$$

where $\hat{u} = \hat{u}(s)$ is the projection of \hat{t} and $\hat{v} = \hat{v}(s)$ is orthogonal to \hat{u} following the right hand rule. Written in vector form, local 3D coordinates $(t, n, b)^t$ project as local 2D coordinates

$$(-\sin\beta t + \cos\beta\cos\alpha n + \cos\beta\sin\alpha b, -\sin\alpha n + \cos\alpha b)^t.$$

The projection of the axis point $A(s)$ is thus the origin of the local 2D frame.

From equations (1) and (7), a point $P(s, \theta)$ on the surface can be shown to project as

$$pr \begin{pmatrix} \cos\beta\cos(\theta - \alpha) \\ \sin(\theta - \alpha) \end{pmatrix} \quad (8)$$

Without loss of generality, from this point, we will normalize the scaling function by fixing $\rho = 1^3$.

In sections 3 and 4 we will derive two types of properties. The first one relates the projection of limb points of the same cross-section (i.e. for the same s along the axis). We call such points *corresponding* limb points. Finding such relationships is useful not only for a *consistent*⁴ 2D description but for recovery of 3D shape as well. This has also been the focus of previous work in [Ponce et al. 1989, Ulupinar & Nevatia 1990a, Ulupinar & Nevatia 1991]. The second type addresses the relationship between the axis of the projection of a Circular PRCG and the projection of its 3D axis. These latter two are not the same in general (Figure 4).

³the scaling function will only change by a constant factor.

To make the discussion in subsequent sections rigorous, we give a number of definitions clarifying our terminology. See Figure 4.

Definition2: a *correspondence segment* is the 2D line segment joining *corresponding* limb points.

Definition3: the *2D axis* (axis of the projection) is the locus of midpoints of correspondence segments.

From equation (8), a correspondence segment \vec{C} , between projections of limb points $P(s, \theta_1), P(s, \theta_2)$, can be expressed in its local 2D frame as

$$r \begin{pmatrix} \cos\beta(\cos(\theta_2 - \alpha) - \cos(\theta_1 - \alpha)) \\ \sin(\theta_2 - \alpha) - \sin(\theta_1 - \alpha) \end{pmatrix} \quad (9)$$

and its midpoint is thus given by

$$\frac{r}{2} \begin{pmatrix} \cos\beta(\cos(\theta_2 - \alpha) + \cos(\theta_1 - \alpha)) \\ \sin(\theta_2 - \alpha) + \sin(\theta_1 - \alpha) \end{pmatrix} \quad (10)$$

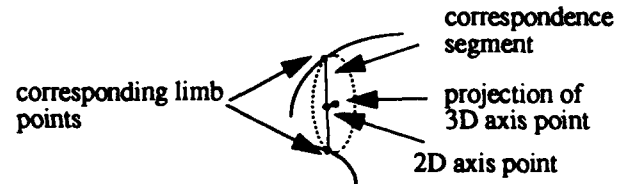


Figure 4 projection of a Circular PRCG

3 Projective invariant properties of special Circular PRCGs

In this section we derive projective *invariant* properties of the two special cases of Circular PRCGs: SORs and Circular PRCGs. Figure 5 illustrates those properties. We will address the general case in section 4.

3.1 Circular PRCGs

Property 1: In an orthographic projection of a Circular PRCG, the 2D axis and the projection of the 3D axis coincide regardless of the viewing direction.

Proof: from equation (6), at limb points, we have $\cos(\theta_1 - \alpha) = \cos(\theta_2 - \alpha) = 0$ and thus $\sin(\theta_1 - \alpha) = -\sin(\theta_2 - \alpha) = \pm 1$, for each s along the axis. Therefore, from equation (10), the 2D axis point is given by $(0, 0)^t$, the origin of the local 2D frame which, as discussed in section 2.2, is the projection of the axis point $A(s)$ no matter what the viewpoint (given by α and β) is \square .

⁴there may be many ways to describe 2D surfaces (such as different classes of ribbons), not all of them being projections of 3D descriptions. For example, points on the same cross-section of a 2D ribbon are not necessarily projections of points on the same 3D cross-section. We call a 2D description *consistent* when it does relate projections of corresponding 3D points (i.e. the 2D description is the projection of the 3D one).

Property 2: In an orthographic projection of a Circular PRCGC, correspondence segments and tangents to the 2D axis at their midpoints are orthogonal.

Proof: reporting the results of the previous analysis in equation (9), we obtain the expression of the correspondence segment $r(0, \pm 2)^t$. The 2D axis shown to be the projection of the 3D axis, its tangent vector is the projection of the 3D tangent vector \hat{t} . This latter, from the projection equation (7), is given by $(-\sin\beta, 0)^t$ which is orthogonal the previous correspondence segment \square .

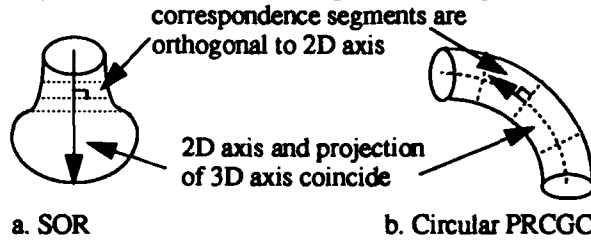


Figure 5 Projective invariants for special Circular PRCGs

Note that it can be easily verified that for a Circular PRCGC, the length of the correspondence segments (in 2-D) is constant and equal to the (constant) diameter of the 3-D cross-section.

3.2 SORs

The following properties are related to the bilateral symmetry property derived by Nalwa in [Nalwa 1989] where non algebraic proofs were used. We state our properties here and give algebraic proofs.

Property 3: In an orthographic projection of a SOR, the 2D axis and the projection of the 3D axis coincide regardless of the viewing direction.

Proof: from equation (5), we have $\cos(\theta_1 - \alpha) = \cos(\theta_2 - \alpha)$ and thus $\sin(\theta_1 - \alpha) + \sin(\theta_2 - \alpha) = 0$. Reporting this in equation (10), the 2D axis point is given by $(r/2)(\cos\beta[\cos(\theta_2 - \alpha) + \cos(\theta_1 - \alpha)], 0)^t$ which is always a point on the u -axis of the local 2D frame; i.e. the direction of the projection of the tangent to the 3D axis. Note that unlike property 1, this point does *not* coincide with the projection of the 3D axis point $A(s)$. However, since the 3D axis is straight, its projection is also straight and it is determined by the origin of the local 2D frame (projection of $A(s)$) and the projection of the 3D tangent; i.e. the u -axis \square .

Property 4: In an orthographic projection of a SOR, correspondence segments and tangents to the 2D axis at their midpoints are orthogonal.

Proof: from the previous proof and equation (9), the correspondence segment is given by $r(0, \sin(\theta_2 - \alpha) - \sin(\theta_1 - \alpha))^t$; i.e. parallel to the v -axis of the local 2D frame. But also from the previous proof the 2D axis is the u -axis, orthogonal to the v -axis \square .

These properties say that in the image plane, the projection of a Circular PRCGC or of a SOR can be consis-

tently described by a ribbon with 2D cross-section (correspondence segments) orthogonal to the (2D) axis tangent (so-called Brooks' ribbon with right angle).

4 Quasi-invariant properties of general Circular PRCGs

In the more general case where the axis is non straight and the cross-section non constant, the limb equation (4) can be rewritten as follows (dropping the argument s and as previously mentioned normalizing the scaling function by fixing $\rho = 1$)

$$(1 - e \cos\theta) \cos(\theta - \alpha) = r \cot\beta \quad (11)$$

where $e = \kappa r = r/R$ (R being the radius of curvature of the axis). e is a local measure of the relative *thickness* of the shape of the Circular PRCGC. Smaller values indicate rather elongated surfaces (small cross-section radius compared to axis curvature radius) whereas larger values thick and curved surfaces. We will call e the *thickness ratio*. r is a measure of how fast the cross-section changes its size (radius).

Equation (11) indicates that two pairs of parameters affect the behavior of the contours of a Circular PRCGC:

- (α, β) corresponding to the viewing direction
- (e, r) corresponding to the object parameters (local shape measures)

For the two sub-cases of Circular PRCGs, the study was simplified by $e = 0$ for SORs and $r = 0$ for Circular PRCGCs (for cylinders, the simplest Circular PRCGC, both are zero). In the general case when both are non zero, the properties discussed previously are *not* invariant. This can easily be seen by expressing the tangent \hat{T}_{2D} to the 2D axis in the general case (omitting the details of its derivation):

$$(0.5 \cos\beta [\dot{r}(c_1 + c_2) - r(\dot{\theta}_1 c_1 + \dot{\theta}_2 c_2)] - \sin\beta (1 - 0.5 \kappa r (\cos\theta_1 + \cos\theta_2)), 0.5(\dot{r}(s_1 + s_2) + r(\dot{\theta}_1 c_1 + \dot{\theta}_2 c_2)))^t \quad (12)$$

where $c_i = \cos(\theta_i - \alpha)$, $s_i = \sin(\theta_i - \alpha)$, $\dot{\theta}_i = \partial\theta_i/\partial s$ for $i = 1, 2$ (at limb points).

Using the expression of the correspondence segment (eq. (9)), the dot product with the 2D axis tangent is given by

$$\langle \vec{C}, \hat{T}_{2D} \rangle = r [0.5 \cos\beta^2 (c_2 - c_1) [\dot{r}(c_1 + c_2) - r(\dot{\theta}_1 c_1 + \dot{\theta}_2 c_2)] - \sin\beta (c_2 - c_1) [1 - 0.5 \kappa r (\cos\theta_1 + \cos\theta_2)] + 0.5(s_2 - s_1)(\dot{r}(s_1 + s_2) + r(\dot{\theta}_1 c_1 + \dot{\theta}_2 c_2))] \quad (13)$$

This expression has been proven to be zero for SORs (where $\kappa = 0$, $\theta_1 = -\theta_2$, $c_1 = c_2$ and $s_1 = -s_2$) and for Circular PRCGCs (where $r = 0$, $c_1 = c_2 = 0$ and $s_1 = -s_2 = \pm 1$), in the previous analysis. In the general case, it is non zero.

No property has been found to be projective invariant for the general case (and we conjecture that none exist). However, we demonstrate that the properties discussed previously are *quasi-invariant* properties of general

Circular PRGCs with respect to both the viewing direction parameters (α, β) (i.e. *orthographic* quasi-invariant) and to (e, \dot{r}) (i.e. *object parameters* quasi-invariant). For this we have to:

- define our space of observation (space of values of the parameters)
- show that the property holds up to some small range over most of that space

4.1 Space of observation

The whole space of observation is a 4-D space $(\alpha, \beta, e, \dot{r})$. For the same object, those parameters vary as s varies. α and β can take any values on the viewing sphere for which the limb equation admits a finite (non empty) set of solutions; i.e. $\alpha \in [0, 2\pi)$ and $\beta \in (0, \pi) \cup (\pi, 2\pi)$. Letting $\alpha \in [0, \pi]$ and $\beta \in (0, \pi)$ is sufficient as the limb equation (11) is symmetric with respect to π for α and β .

The 2D subspace (e, \dot{r}) is also constrained. We have $|e| \leq 1$ (the cross-section radius is smaller than the radius of curvature of the axis), otherwise as mentioned previously, the surface self-intersects. \dot{r} is also constrained, since $|1 - e \cos \theta| \leq 2$ (as $|e| \leq 1$) and thus, from equation (11), $|\dot{r}| \leq 2|\tan \beta|$. This implies that, at each point, the closer the viewing direction is to the axis tangent direction (i.e. smaller $|\tan \beta|$), the smaller $|\dot{r}|$ has to be (otherwise there would be no visible limbs). Thus, for a surface point where $\beta = 15^\circ$, for example, the cross-section has limbs if $|\dot{r}| < 0.53$. Objects seen in daily environments, such as animal limbs or industrial parts, appear not to have, at the same time, high values of e and \dot{r} ; i.e. when e is high $|\dot{r}|$ is small (otherwise the thickness ratio would rapidly increase, which would cause self-intersection) and when $|\dot{r}|$ is high e is small. Shapes with high thickness ratio and high sweeping slope self-occlude over most viewing directions. In the analysis, values of $|\dot{r}|$ will be given as rates of change of the cross-section radius r per unit arclength along the axis.

4.2 Quasi-invariant properties

Property 5: In an orthographic projection of a Circular PRGC, orthogonality of correspondence segments and tangents to the 2D axis at their midpoints is a *quasi-invariant* property (the angle they form is "almost" right over most of the space of observation) with respect to the viewing direction (α, β) and object parameters (e, \dot{r}) .

Note that this property is related to the description of the projection of a Circular PRGC by a Brooks's ribbon with right angle.

To prove property 5 and analyze to what extent the 2D angle is "almost" right, we analyze the behavior of the angle between a correspondence segment and the tangent to the 2D axis for all parameter values in our space

of observation. (Notice that in 3D, the segment connecting limb points is orthogonal to the tangent to the 3D axis since the cross-section is planar and orthogonal to the axis). Unlike an invariant property, algebraic analysis of a quasi-invariant property is difficult. Instead, we analyze it numerically by quantizing the space of observation and, for each point $(\alpha, \beta, e, \dot{r})$ of the space, solving the limb equation and deriving the projections of corresponding limb points and the tangent to the 2D axis. Although there exists an analytical expression for the tangent to the 2D axis (eq. (12)), it requires knowledge of how θ varies with respect to s at limb points, which is not known in general. Instead, we have used the following method (of which, we omit the details):

for each set of parameters $(\alpha_1, \beta_1, e_1, \dot{r}_1)$ (at some s) do

- select an arbitrary 3D frame $F_1 = (\hat{i}_1, \hat{n}_1, \hat{b}_1)$
- solve the limb eq. (11) to obtain the two limb points θ_{11} and θ_{12}
- determine $(\alpha_2, \beta_2, e_2, \dot{r}_2)$ at $s+ds$ (for some small ds)
- solve eq. (11) for the second pair of limb points θ_{21} and θ_{22} (at $s+ds$) and express their coordinates in F_1
- using eq. (8) determine the projection of the two pairs of points $P(s, \theta_{11})$, $P(s, \theta_{12})$ and $P(s+ds, \theta_{21})$, $P(s+ds, \theta_{22})$ (say p_{11} , p_{12} , p_{21} and p_{22})
- determine the angle between the correspondence segment given by p_{11} and p_{12} and the 2D axis tangent given by the line joining the midpoints of $p_{11} p_{12}$ and $p_{21} p_{22}$.

We have derived the angles in the image between correspondence segments and 2D axis tangents over the space of observation defined by $\alpha \in [0, \pi]$, $\beta \in (0, \pi)$, $e \leq 0.5$ and $|\dot{r}| < 0.5$ (sweep rates less than half the current cross-section radius per unit arclength). The results show that over 84.30% of that 4D space (excluding special values $e = 0$ or $\dot{r} = 0$; i.e. SORs and Circular PRGCs) the 2D angle is within 5° of 90° and for over 92.63% of the space is within 10° of 90° . Table 1 summarizes the sizes (in percent) of the regions on the viewing sphere where the 2D angle is within 5° of 90° for certain values of (e, \dot{r}) . The size is with respect to the space region where limbs exist. (It can be seen, from this table, that the size of the space where the property holds (2D angle within 5° of being right) gradually decreases as both e and \dot{r} take higher values). Figure 6a shows a 3D plot of the 2D angle as a function of (α, β) for $(e, \dot{r}) = (0.2, 0.2)$ and Figure 6b shows the corresponding display of the half viewing sphere $((\alpha, \beta)$ sub-space). This last figure shows where the property holds, where it does not hold and where limbs do not ex-

ist. In the display, vertical circles correspond to constant β values with a 5° step.

	$e = 0.1$	$e = 0.2$	$e = 0.3$	$e = 0.4$
$\dot{r} = 0.1$	96.72	92.44	88.50	85.57
$\dot{r} = 0.2$	96.86	89.44	83.62	78.55
$\dot{r} = 0.3$	97.11	88.25	80.05	74.83
$\dot{r} = 0.4$	97.29	87.57	78.72	72.86

Table 1 Viewing sets sizes (percent) where the observed 2D angle is within 5° of 90°

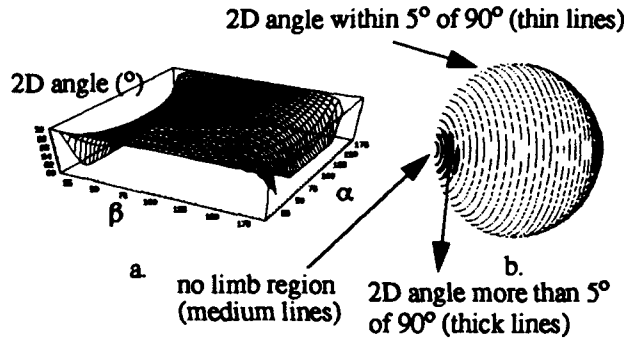


Figure 6 Property 5 for $(e, \dot{r}) = (0.2, 0.2)$.
a. 3D plot. b. half viewing sphere.

Notice that the sub-space where the property holds is connected and the property is well behaved. It tends to gradually degrade for small values of $|\tan\beta|$, that is close to regions where limbs do not exist. Notice that a small value of $|\tan\beta|$ requires a very specific viewpoint where the viewing direction \hat{V} is not only close to being in the axis plane but also almost parallel to the 3D axis tangent; i.e. an unlikely viewpoint. Therefore, even if \hat{V} is close to being in the axis plane, the property would degrade only at points where the axis tangent is in the direction of \hat{V} (a small set of points) and it would still hold for the rest of the surface (a much larger set).

Property 6: In an orthographic projection of a Circular PRGC, the tangent to the 2D axis and the tangent to the projection of 3D axis for corresponding points are "almost" parallel (within a few degrees of each other) over a large fraction of the space of observation.

To show this property and analyze the extent to which the two 2D tangent vectors are "almost" parallel, we have used the method discussed previously for property 5. The results showed that the two tangents are within 3° of each other over 94.48% of the previous space of observation and within 5° over 96.90% of that space. Table 2 gives the sizes of the regions on the viewing sphere where the two tangents are within 3° of each other for some values of (e, \dot{r}) . Figure 7a gives the 3D plot of the angle difference as a function of (α, β) for $(e, \dot{r}) = (0.2, 0.2)$ and Figure 7b the corresponding half viewing sphere, such as discussed for Figure 6b. The behavior of this quasi-invariant property is similar to the previous one. It tends to degrade only close to re-

gions where limbs do not exist (for unlikely viewpoints). Notice that because the size of the regions on

	$e = 0.1$	$e = 0.2$	$e = 0.3$	$e = 0.4$
$\dot{r} = 0.1$	96.80	95.04	92.26	91.76
$\dot{r} = 0.2$	99.21	95.45	93.01	90.78
$\dot{r} = 0.3$	100	97.31	94.27	92.82
$\dot{r} = 0.4$	100	99.00	96.30	94.07

Table 2 Viewing sets sizes (percent) where the tangent to the 2D axis is within 3° of the tangent to the projection of the 3D axis

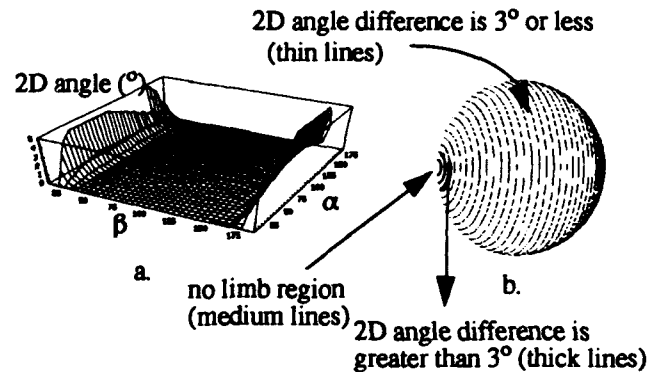


Figure 7 Property 6 for $(e, \dot{r}) = (0.2, 0.2)$.
a. 3D plot. b. half viewing sphere.

the viewing sphere where limbs do not exist is mainly influenced by \dot{r} (as discussed in section 4.1), the relative size of the region where the property holds may tend to increase as \dot{r} increases (the ratio is over a smaller region, where limbs exist at all).

Notice that for SORs and Circular PRGCs this property is a projective *invariant* as we have proved the stronger property that the two axes coincide. For Circular PRGCs the tangents are the same since the axes coincide precisely at corresponding points. For SORs, the axes are collinear (although they do *not* coincide at corresponding points), thus they have parallel tangents.

Notice also that it can be verified from equations (10) and (13), that the properties hold exactly (perfect orthogonality and coincidence of axes) for general Circular PRGCs for the two viewpoints, where the viewing direction \hat{V} is orthogonal to the axis plane (i.e. side view, $\cos\alpha = \cos\beta = 0$, $\theta_1 = 0$, $\theta_2 = \pi$) and where it is in the axis plane (i.e. frontal view, $\sin\alpha = 0$, $\cos\theta_1 = \cos\theta_2$, $\sin\theta_1 = -\sin\theta_2$).

These two properties are quasi-invariant with respect to transforms parameterized over the 4D space of observation. Thus they are orthographic quasi-invariants (with respect to (α, β)) and object-shape parameters quasi-invariants (with respect to (e, \dot{r})). Usage of these properties is discussed in the next section.

5 Application to shape description and recovery

In this section we discuss how the projective properties of the previous section can be used to derive constraints on shape recovery for Circular PRGCs. In section 5.1 we discuss the usage of property 5 for a 2D consistent shape description. In section 5.2 we discuss how constraints derived from both the 2D description (property 5) and property 6 can be used to recover the 3D shape of Circular PRGCs from a single image of their contours. The recovery method provides tests as to whether the contours are projections of Circular PRGCs, thus it does not make assumptions about the class of objects being viewed. At this stage, we are assuming perfect contours to be given as our purpose is to address usage of the new projective quasi-invariants. Our ultimate goal is to handle real image contours. Previous experience with SHGCs shows that projective invariants help solve the figure-ground problem in complex real images [Zerroug & Nevatia 1993]. We plan on implementing a similar approach for handling Circular PRGCs using the derived quasi-invariants, in the near future.

5.1 2D shape description

Property 5 indicates that the projection of a Circular PRGC can be described by a ribbon whose cross-sections (correspondence segments) are orthogonal to the (curved) axis. This type of ribbon has been addressed in the past by many researchers. Nevatia and Binford [Nevatia & Binford 1977] used it to describe complex objects from contours obtained from range data. In AC-RONYM [Brooks 1983], Brooks used these ribbons in a model based interpretation of image contours and Rao and Nevatia [Rao & Nevatia 1989] used them to describe complex shapes from imperfect contours. In [Ponce 1988], Ponce compared those ribbons (Brooks' ribbons⁵) to other types of ribbons commonly used in the literature. We will call the ribbons in our analysis *right ribbons*.

In such previous work, right ribbons have been used as a means for 2D shape description without rigorously addressing the relationship between the obtained 2D descriptions and the 3D descriptions of viewed objects. In this discussion, we have shown that right ribbons are indeed good descriptors of the orthographic projection of Circular PRGCs. They provide a consistent 2D description by identifying corresponding limb points, projections of points on the same cross-section of the 3D shape.

Detection of right ribbons has been addressed in [Nevatia & Binford 1977, Rao & Nevatia 1989]. The

⁵Ribbons in our analysis are a special case of Brooks' ribbons with a right angle between cross-sections and axis tangents.

original method (projection method) consists of discretizing the orientations of the axis (or equivalently of the correspondence segments) and casting regularly spaced lines. We have used a variation of that method using a quadratic B-spline representation of the contours of Circular PRGCs. Line casting is done from the extremities of each B-spline segment. The complexity of this process of $O(km)$ where k is the number of selected orientations and m the number of B-spline segments. In the original method, m is the number of points, which is much larger than the number of B-spline segments. *Local right ribbons* are detected for each pair of lines intersecting a pair of curves at nearby B-splines (Figure 8). The local axis can be determined *analytically* given the desired extremities m_0 and m_n and their tangents \vec{T}_0 and \vec{T}_n . A local right ribbon is hypothesized if there exists a quadratic B-spline segment (local axis) for those extremities and the following constraint is satisfied (see Figure 8):

$dist(p_a, p_m) / dist(p_i, q_i) < \epsilon \quad \forall i$; where p_a is a point on the local axis, p_i, q_i hypothesized corresponding points (intersections of the line orthogonal to the axis at p_a with opposite B-splines) and p_m the midpoint of $p_i q_i$; i.e the local axis should be the locus of midpoints of correspondence segments.

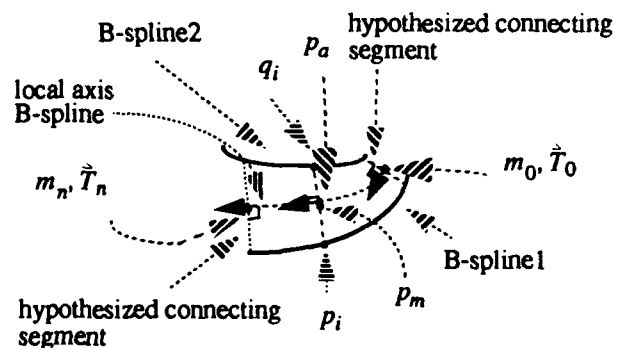


Figure 8 Local right ribbons detection using B-splines.

Several such local right ribbons are usually obtained between each pair of curves. To obtain a global description (of the whole surface), we perform a grouping of these local ribbons based on contiguity of their sides and their orientations. Selection of the 'best' global description uses measures of continuity of orientations of correspondence segments and axes. For lack of space, we omit further details of the method as they are similar to [Rao & Nevatia 1989]. Figure 9, shows the obtained right ribbon descriptions for the two Circular PRGCs of Figure 1 using the above method.

5.2 3D shape recovery

The 2D descriptions can be used to recover complete 3D object centered descriptions from 2D contours of Circular PRGCs. To do this, we have to recover the 3D cross-section, the 3D axis and the sweeping function.

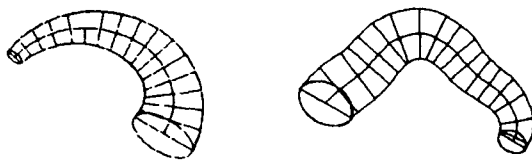


Figure 9 Resulting right ribbon descriptions for Circular PRGCs of Figure 1.

For this, we assume that the viewpoint is general and that the two extremal cross-sections are (at least partially) visible. Since the cross-section is known to be circular, recovery of the two extremal 3D cross-sections can be done by finding the orientation of the 3D circles, (C_1 and C_2 in Figure 10), whose projections coincide with the visible elliptic cross-sections⁶. The 3D circle finding method also yields the 3-D orientations \hat{n}_1 and \hat{n}_2 of the two extremal 3D cross-sections. The orientation of the 3D axis plane is, thus, $\hat{n}_a = \hat{n}_1 \times \hat{n}_2$. The depth of the axis plane can be arbitrarily fixed. The 3D positions of the two extremal cross-sections are automatically fixed since their centers belong to the axis plane.

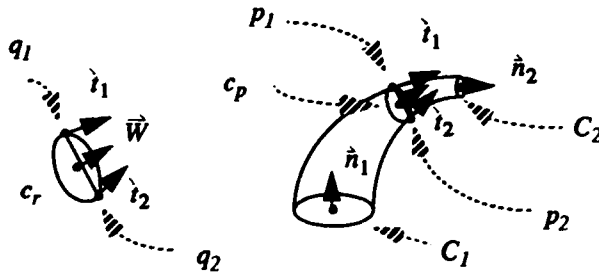


Figure 10 Recovering 3D models of Circular PRGCs from 2D contours

As previously mentioned, the 3D axis is not the back-projection of the 2D axis on the axis plane, as property 6 relates only 2D tangent orientations, not axis points. However, the 2D description, based on property 5, gives us a good approximation of corresponding points (projections of points on the same 3D cross-section; p_1 and p_2 in Figure 10). Moreover, property 6 gives us a good approximation of the orientation of the projection, say \vec{w} , of the 3D axis tangent (3D cross-section orientation). The (unique) 3D cross-section whose projection passes through p_1 and p_2 can be determined as follows (see Figure 10):

- Backproject \vec{w} on the 3D axis plane to obtain the orientation of the 3D cross-section, say \vec{w} .

⁶of the two possible solutions for each ellipse, we choose the one that makes the exterior part (outside the ribbon) point away from the viewer. In the case of a visible half ellipse, the part of the 3D circle corresponding to the visible half ellipse is made closer to the viewer than the one corresponding to the invisible (occluded) part.

- In 3D, rotate a reference extremal cross-section, say C_1 , by the rotation $R(\hat{n}_1, \vec{w})$ which rotates \hat{n}_1 to \vec{w} . Call the resulting circle C_r and its projection c_r .
- Find, on c_r , the two points whose tangents are the same as the tangents \vec{t}_1 and \vec{t}_2 to the projected limbs at p_1 and p_2 ⁷. Call them q_1 and q_2 .
- Since lengths ratio of parallel segments is an orthographic invariant, the scale of the desired cross-section with respect to the reference one is given by $r = \text{dist}(p_1, p_2) / \text{dist}(q_1, q_2)$.
- Scale c_r by r and translate it so that p_1 and q_1 coincide. Call the resulting cross-section c_p .
- The desired 3D cross-section is the backprojection, C_p , of c_p so that its center backprojects on the 3D axis plane. This gives us the 3D cross-sections (thus, the 3D sweep as well).
- The 3D axis is the locus of the centers of the 3D cross-sections so recovered.

Results of the application of this 3-D recovery method to the descriptions of Figure 9 are shown in Figure 11. The figure shows the 3-D ruled surfaces, showing cross-sections, meridians and the 3-D axes, and the corresponding shaded displays, using different poses of the recovered 3-D shapes

This method produces estimated 3-D shapes from the observed 2D contours, since the determined 2-D correspondences are themselves estimates of the actual ones. However, the derived quasi-invariants show that the projections of the 3-D correspondences and the estimated 2-D correspondences are close to each other *over most viewing directions*. Therefore, the (unique) back-projections of those 2-D correspondences are close to the actual 3-D ones; i.e. the recovered 3-D shapes are good approximations of actual ones. Notice also that the method does not make assumptions about the viewed objects. It provides tests as to whether the objects are projections of Circular PRGCs. The tests involve consistency of orientations of extremal cross-sections as predicted by the 2-D correspondences and as determined by the 3-D circle finding method. The former orientations are determined by the backprojections of the 2-D axis tangents, at the extremities of the surface, on the axis plane. The latter ones are the normals \hat{n}_1 and \hat{n}_2 mentioned previously. Other objects would produce inconsistencies in those orientations as the extremities of the limbs would not be matched by the right ribbon finding method (we have found that non Circular PRGCs do not satisfy the properties 5 and 6 as Circular PRGCs do).

⁷in the projection, cross-sections and limbs are mutually tangential, and vector parallelism is an orthographic invariant.

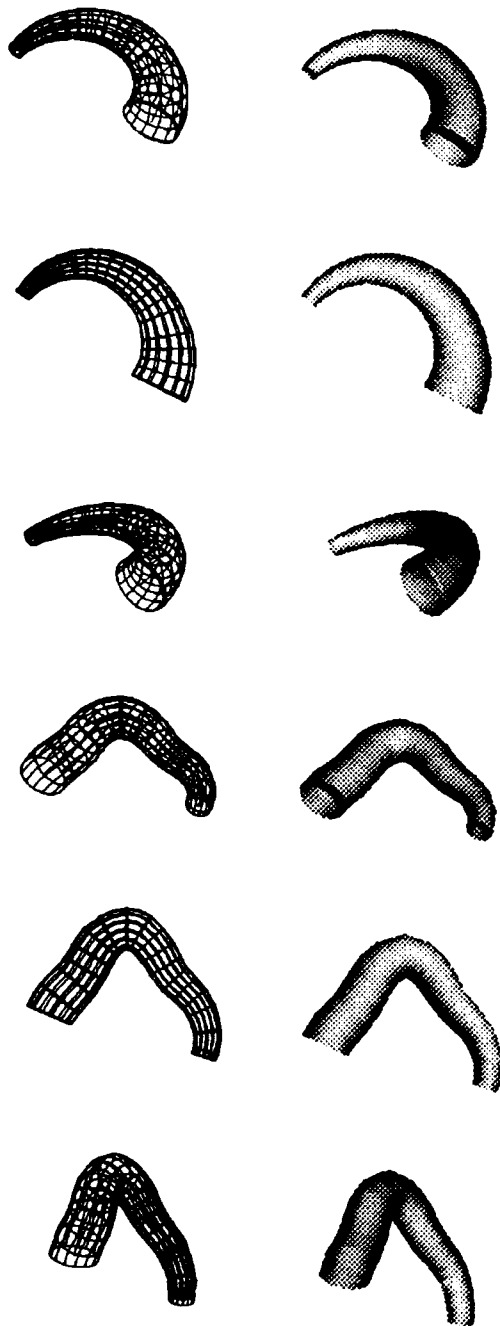


Figure 11 Results of 3D shape recovery for the previous two Circular PRGCs.

6 Conclusion

We have addressed projective properties of the contours of complex GC primitives (Circular PRGCs). Previous work on using projective properties of GCs for interpretation of 2D contours has addressed relatively simple primitives having either a straight axis or a constant size cross-section, where *invariant* properties have been derived and used for 3D shape recovery. In the

case of Circular PRGCs (non straight axis and non constant cross-section size), we have derived rigorous *quasi-invariant* properties for the general case and have shown that they are invariant for special cases. Those quasi-invariants have been derived by analyzing the behavior of limb projections as a function of viewing parameters and object shape parameters. The derived quasi-invariants provide strong constraints for consistent 2D descriptions and 3D shape recovery from 2D contours and we have demonstrated their application on some examples. We believe that the results derived in this analysis constitute an important progress towards handling complex objects with non simple primitives.

In the near future, we plan on addressing recovery of Circular PRGCs from real image contours. Other efforts have shown that projective invariants help solve the figure ground problem for SHGCs in real image contours with breaks, markings, and occlusion [Zerroug & Nevatia 1993]. We believe that quasi-invariants are also a source of strong constraints for the figure ground problem for non simple GC primitives. Our aim is to develop a system that handles compound objects made up of several GC primitives.

References

- [Biederman 1987] I. Biederman, "Recognition by components: A theory of human image understanding," *Psychological Review*, 94(2):115-147, 1987.
- [Binford 1971] T.O. Binford, "Visual perception by computer," *IEEE Conference on Systems and Controls*, December 1971, Miami.
- [Binford 1981] T.O. Binford, "Inferring surfaces from images," *Artificial Intelligence*, 17:205-245, 1981.
- [Binford et al. 1987] T.O. Binford, T.S. Levitt and W.B. Mann, "Bayesian inference in model-based machine vision," *Proceedings of AAAI Uncertainty Workshop*, 1987.
- [Binford 1991] T.O. Binford, "Inverse generalized cylinders: Inverse generalized Transformational Invariance and Quasi-Invariance," In *Proceedings of Darpa-Esprit Workshop on Invariance in Computer Vision*, 1991.
- [Brooks 1983] R.A. Brooks, "Model-based three dimensional interpretation of two dimensional images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):140-150, 1983.
- [Clowes 1971] M.B. Clowes, "On seeing things," *Artificial Intelligence*, 2(1):79-116, 1971.
- [Forsyth et al. 1991] D.A. Forsyth, J.L. Mundy, A.P. Zisserman, C. Coelho, A. Heller and C.A. Rothwell, "Invariant Descriptors for 3-D Object Recognition and Pose," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:971-991, 1991.

- [Gross & Boulton 1990] A. Gross and T. Boulton, "Recovery of generalized cylinders from a single intensity view," In *Proceedings of the Image Understanding Workshop*, pages 557-564, Pennsylvania, 1990.
- [Heraud & Brady 1988] R. Heraud and M. Brady, "On the geometric interpretation of image contours," *Artificial Intelligence*, 37:333-353, 1988.
- [Kanade 1981] T. Kanade, "Recovery of the three-dimensional shape of an object from a single view," *Artificial Intelligence*, 17:409-460, 1981.
- [Koenderink 1990] J.J. Koenderink, "Solid Shape," M.I.T. Press, Cambridge, MA, 1990.
- [Mackworth 1973] A.K. Mackworth, "Interpreting pictures of polyhedral scenes," *Artificial Intelligence*, 4:121-137, 1973.
- [Malik 1987] J. Malik, "Interpreting line drawings of curved objects," *International Journal of Computer Vision*, 1(1):73-103, 1987.
- [Marr 1977] D. Marr, "Analysis of occluding contour," In *Proceedings of the Royal Society of London*, vol B197, 441-475, 1977.
- [Millman & Parker 1977] R.S. Millman and G.D. Parker, "Elements of differential geometry," Prentice Hall, 1977.
- [Nalwa 1989] V. Nalwa, "Line drawing interpretation: Bilateral symmetry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:1117-1120, 1989.
- [Nevatia & Binford 1977] R. Nevatia and T.O. Binford, "Description and recognition of complex curved objects," *Artificial Intelligence*, 8(1):77-98, 1977.
- [Ponce & Chelberg 1987] J. Ponce and D. Chelberg, "Finding the limbs and cusps of generalized cylinders," *International Journal of Computer Vision*, 1:195-210, 1987.
- [Ponce 1988] J. Ponce, "Ribbons, Symmetries and Skewed Symmetries," In *Proceedings of the Image Understanding Workshop*, pages 1074-1079, Massachusetts, 1988.
- [Ponce et al. 1989] J. Ponce, D. Chelberg and W.B. Mann, "Invariant properties of straight homogeneous generalized cylinders and their contours," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(9):951-966, 1989.
- [Rao & Medioni 1988] K. Rao and G. Medioni, "Useful geometric properties of the generalized cone," In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 276-281, 1988.
- [Rao & Nevatia 1989] K. Rao and R. Nevatia, "Description of complex objects from incomplete and imperfect data," In *Proceedings of the Image Understanding Workshop*, pages 399-414, Palo Alto, California, May 1989.
- [Richetin et al. 1991] M. Richetin, M. Dhome, J.T. Laprestre and G. Rives, "Inverse Perspective Transform Using Zero-Curvature Contours Points: Applications to the Localization of Some Generalized Cylinders from a Single View," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(2):185-192, 1991.
- [Saint-Marc & Medioni 1990] P. Saint-Marc and G. Medioni, "B-spline contour representation and symmetry detection," In *First European Conference on Computer Vision*, pages 604-606, Antibes, France, April 1990.
- [Sato & Binford 1992] H. Sato and T.O. Binford, "BUILDER-I: a system for the extraction of SHGC objects in an edge image," In *Proceedings of the Image Understanding Workshop*, San Diego, California, January 1992.
- [Shafer & Kanade 1983] S.A. Shafer and T. Kanade, "The theory of straight homogeneous generalized cylinders," Technical Report CS-083-105, Carnegie Mellon University, 1983.
- [Ulupinar & Nevatia 1990a] F. Ulupinar and R. Nevatia, "Shape from contours: SHGCs," In *Proceedings of IEEE International Conference on Computer Vision*, pages 582-582, Osaka, Japan, 1990.
- [Ulupinar & Nevatia 1990b] F. Ulupinar and R. Nevatia, "Inferring shape from contours for curved surfaces," In *Proceedings of International Conference on Pattern Recognition*, pages 147-154, Atlantic City, NJ, 1990.
- [Ulupinar & Nevatia 1991] F. Ulupinar and R. Nevatia, "Recovering Shape from Contour for Constant Cross Section Generalized Cylinders," In *Proceedings of Computer Vision and Pattern Recognition*, pages 674-676, 1991, Maui, Hawaii.
- [Zerroug & Nevatia 1993] M. Zerroug and R. Nevatia, "Scene segmentation and volumetric descriptions of SHGCs from a single intensity image," In *Proceedings of the Image Understanding Workshop*, Washington DC., 1993.

Invariants of Lines in Space

Richard Hartley

GE - Corporate Research and Development,
P.O. Box 8, Schenectady, NY, 12301.

Ph : (518)-387-7333

Fax : (518)-387-6845

email : hartley@crd.ge.com

Abstract

This paper describes a pair of projectivity invariants of four lines in three dimensional projective space, \mathcal{P}^3 . Invariants are derived in both algebraic and geometric terms, and the connection between the two ways of defining the invariants is established. Since a count of the number of degrees of freedom would predict the existence of a single invariant, rather than the two that are shown to exist, an isotropy of the four lines must exist. The nature of this isotropy is investigated.

It is shown that once the epipolar geometry is known, the invariants of four lines may be computed from the images of the four lines in two distinct views with uncalibrated cameras. An example with real images is computed to show that the invariants are effective in distinguishing different geometrical configurations of lines.

1 Introduction

Projective invariants of geometrical configurations in space have recently received much attention because of their application to vision problems ([Mundy-Zisserman-92]). Although invariants of a wide range of objects in the 3-dimensional projective space \mathcal{P}^3 do exist ([Abhyankar-92]), one is restricted in vision applications to considering those that may be computed from two-dimensional projections (images). For point sets and more structured geometrical objects lying in planes in \mathcal{P}^3 , many invariants exist ([Coelho-92]) which can be computed from a single view. Unfortunately, it has been shown in [Burns-92] that no invariants of arbitrary point sets in 3-dimensions may be computed from a single image. One is led either to consider constrained sets of points, or else to allow two independent views of the object. An example of the first approach is contained in [Zisserman-92] which considers solids of revolution. This paper takes the second course and considers invariants that can be derived from two views of an object. It has been recently proven in [Faugeras-92] and [Hartley-Gupta-92] that a 3 dimensional scene may be constructed up to a projectivity of space from two views with uncalibrated cameras. This allows us to compute invariants of 3-dimensional configurations from two views. Invariants of six points in space have been suggested in [Faugeras-92] and

[Hartley-Gupta-92] and verified in [Hartley-93a] to be useful at distinguishing different point configurations. The present paper considers invariants of straight lines in \mathcal{P}^3 computable from a pair of images. Since straight lines occur commonly in man-made objects and may be effectively extracted from the image using an edge extraction algorithm, invariants of sets of lines may prove to be more useful than invariants of point sets in object recognition applications.

The invariants of lines in space can not be computed from two views of lines only. It may be seen that virtually no information about the cameras can be derived from two views of a set of lines in space. This is because given two images of a line and two arbitrary cameras, there is always a line in space that corresponds to the two images. In other words, two images of an unknown line do not in any way constrain the cameras. This point is discussed in [Weng-92]. If on the other hand the epipolar geometry of the two views (as expressed in the essential matrix) is known, then the locations of lines may be determined up to a projectivity of \mathcal{P}^3 from their images in the two views. There are many ways of determining the epipolar geometry from views of points or lines in two or three images ([Higgins-81, Hartley-93b, Zisserman-Hartley-93]).

2 Line Invariants

In this section, invariants of lines in space will be described. It will be shown that four lines in the 3-dimensional projective plane, \mathcal{P}^3 give rise to two independent invariants under projectivity of \mathcal{P}^3 . Two different ways of defining invariants will be described, one algebraic and one geometric.

2.1 Algebraic Invariant Formulation

Consider four lines λ_i in space. A line may be given by specifying either two points on the line or dually, two planes that meet in the line. It does not matter in which way the lines are described. For instance, in the formulae (2) and (3) below certain invariants of lines are defined in terms of pairs of points on each line. The same formulae could be used to define invariants in which lines are represented by specifying a pair of planes that meet along the line. Since the method of determining lines in space from two view given in section 3.3 gives a representation of the line as an in-

tersection of two planes, the latter interpretation of the formulae is most useful.

Nevertheless, in the following description, of algebraic and geometric invariants of lines, lines will be represented by specifying two points, since this method seems to allow easier intuitive understanding. It should be borne in mind, however, that the dual approach could be taken with no change whatever to the algebra, or geometry.

In specifying lines, each of two points on the line will be given as a 4-tuple of homogeneous coordinates, and so each line λ_i is specified as a pair of 4-tuples

$$\lambda_i = ((a_{i1}, a_{i2}, a_{i3}, a_{i4})(b_{i1}, b_{i2}, b_{i3}, b_{i4}))$$

Now, given two lines λ_i and λ_j , one can form a 4×4 determinant, denoted by

$$|\lambda_i \lambda_j| = \det \begin{pmatrix} a_{i1} & a_{i2} & a_{i3} & a_{i4} \\ b_{i1} & b_{i2} & b_{i3} & b_{i4} \\ a_{j1} & a_{j2} & a_{j3} & a_{j4} \\ b_{j1} & b_{j2} & b_{j3} & b_{j4} \end{pmatrix}. \quad (1)$$

Finally, it is possible to define two independent invariants of the four lines by

$$I_1(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = \frac{|\lambda_1 \lambda_2| |\lambda_3 \lambda_4|}{|\lambda_1 \lambda_3| |\lambda_2 \lambda_4|} \quad (2)$$

and

$$I_2(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = \frac{|\lambda_1 \lambda_2| |\lambda_3 \lambda_4|}{|\lambda_1 \lambda_4| |\lambda_2 \lambda_3|}. \quad (3)$$

It is necessary to prove that the two quantities so defined are indeed invariant under projectivities of \mathcal{P}^3 . First, it must be demonstrated that the expressions do not depend on the specific formulation of the lines. That is, there are an infinite number of ways in which a line may be specified by designating two points lying on it, and it is necessary to demonstrate that choosing a different pair of points to specify a line does not change the value of the invariants. To this end, suppose that $(a_{i1}, a_{i2}, a_{i3}, a_{i4})^T$ and $(b_{i1}, b_{i2}, b_{i3}, b_{i4})^T$ are two distinct points lying on a line λ_i , and that $(a'_{i1}, a'_{i2}, a'_{i3}, a'_{i4})^T$ and $(b'_{i1}, b'_{i2}, b'_{i3}, b'_{i4})^T$ are another pair of points lying on the same line. Then, there exists a 2×2 matrix D_i such that

$$\begin{pmatrix} a'_{i1} & a'_{i2} & a'_{i3} & a'_{i4} \\ b'_{i1} & b'_{i2} & b'_{i3} & b'_{i4} \end{pmatrix} = D_i \begin{pmatrix} a_{i1} & a_{i2} & a_{i3} & a_{i4} \\ b_{i1} & b_{i2} & b_{i3} & b_{i4} \end{pmatrix}.$$

Consequently,

$$\begin{pmatrix} a'_{i1} & a'_{i2} & a'_{i3} & a'_{i4} \\ b'_{i1} & b'_{i2} & b'_{i3} & b'_{i4} \\ a'_{j1} & a'_{j2} & a'_{j3} & a'_{j4} \\ b'_{j1} & b'_{j2} & b'_{j3} & b'_{j4} \end{pmatrix} = \begin{pmatrix} D_i & 0 \\ 0 & D_j \end{pmatrix} \begin{pmatrix} a_{i1} & a_{i2} & a_{i3} & a_{i4} \\ b_{i1} & b_{i2} & b_{i3} & b_{i4} \\ a_{j1} & a_{j2} & a_{j3} & a_{j4} \\ b_{j1} & b_{j2} & b_{j3} & b_{j4} \end{pmatrix}.$$

Taking determinants, it is seen that the net result of choosing a different representation of the lines λ_i and λ_j is to multiply the value of $|\lambda_i \lambda_j|$ by a factor $\det(D_i) \det(D_j)$. Since each of the lines λ_i appears in both the numerator and denominator of the expressions (2) and (3), the factors will cancel and the values of the invariants will be unchanged.

Next, it is necessary to consider the effect of a change of projective coordinates. If H is a 4×4 invertible matrix representing a coordinate transformation of \mathcal{P}^3 , then it may be applied to each of the points used to designate the four lines. The result of applying this transformation is to multiply the determinant $|\lambda_i \lambda_j|$ by a factor $\det(H)$. The factors on the top and bottom cancel, leaving the values of the invariants (2) and (3) unchanged. This completes the proof that I_1 and I_2 defined by (2) and (3) are indeed projective invariants of the set of four lines.

An alternative invariant may be defined by

$$I_3(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = \frac{|\lambda_1 \lambda_4| |\lambda_2 \lambda_3|}{|\lambda_1 \lambda_3| |\lambda_2 \lambda_4|}. \quad (4)$$

It is easily seen, that $I_3 = I_1/I_2$. However, if $|\lambda_1 \lambda_2|$ vanishes, then both I_1 and I_2 are zero, but I_3 is in general non-zero. This means that I_3 can not always be deduced from I_1 and I_2 . A preferable way of defining the invariants of four lines is as a homogeneous vector

$$I(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = (|\lambda_1 \lambda_2| |\lambda_3 \lambda_4|, |\lambda_1 \lambda_3| |\lambda_2 \lambda_4|, |\lambda_1 \lambda_4| |\lambda_2 \lambda_3|) \quad (5)$$

Two such computed invariant values are deemed equal if they differ by a scalar factor. Note that this definition of the invariant avoids problems associated with vanishing or near-vanishing of the denominator in (2) or (3).

The definitions of I_1 and I_2 are similar to the definition of the cross-ratio of points on a line. It is well known that for four points on a line, there is only one independent invariant. It may be asked whether I_1 may be obtained from I_2 by some simple arithmetic combination. This is not the case, as will become clearer when the connection of these algebraic invariants with geometric invariants is shown.

2.2 Degenerate Cases

The determinant $|\lambda_i \lambda_j|$ as given in (1) will vanish if and only if the four points involved are coplanar, that is, exactly when the two lines are coincident (meet in space). If all three components of the vector $I(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ given by (5) vanish, then the invariant is undefined. Enumeration of cases indicates that there are two essentially different configurations of lines in which this occurs.

1. Three of the lines lie in a plane.
2. One of the lines meets all the other three.

The configuration where one line meets two of the other lines is not degenerate, but does not lead to very much useful information, since two of the components

of the vector vanish. Up to scale, the last component may be assumed to equal 1, which means that two such configurations can not be distinguished. In fact any two such configurations are equivalent under projectivity.

2.3 Geometric Invariants of Lines

Consider four lines λ_i in general position (which means that they are not coincident) in \mathcal{P}^3 . It will be shown that there exist exactly two further lines τ_1 and τ_2 , called *transversals*, which meet each of the four lines. Once this is established, it is easy to define invariants.

The points of intersection of each of the four lines λ_i with one of the transversals τ_j constitute a set of four points on a line in \mathcal{P}^3 . The cross ratio of these points is an invariant of the four lines λ_i . In this way, two invariants may be defined, one for each of the two transversals.

Invariants may be defined in a dual manner as follows. Given a transversal, τ_j , meeting each of the lines λ_i , there exists, for each λ_i a plane denoted $\langle \tau_j, \lambda_i \rangle$, containing τ_j and λ_i . This gives rise to a set of four planes meeting in a common line τ_j . The cross-ratio of this set of planes is an invariant of the lines λ_i .

It is easy to see that this dual construction does not give rise to any new invariant. Specifically, consider the cross-ratio of the four planes meeting at τ_1 . The cross-ratio of four planes meeting along a line is equal to the cross-ratio of the points of intersection of the planes with any other non-coincident line in space. The line τ_2 is such a line. Hence, the cross ratio of the planes $\langle \tau_1, \lambda_i \rangle$ is equal to the cross-ratio of the points $\langle \tau_1, \lambda_i \rangle \cap \tau_2$, where the symbol \cap denotes the point of intersection. However, plane $\langle \tau_1, \lambda_i \rangle$ meets τ_2 in the point $\lambda_i \cap \tau_2$. In other words, the cross-ratio of the four planes meeting along τ_1 is equal to the cross-ratio of the four points along τ_2 , and vice-versa.

2.4 Existence of Transversals

To prove the existence of transversals, we start by considering three lines in space.

Lemma 1. *There exists a unique quadric surface containing three given lines λ_1, λ_2 and λ_3 in general position in \mathcal{P}^3 .*

Proof. For a reference to properties of quadric surfaces, the reader is referred to [Semple-Kneebone-52]. It is shown there that a quadric surface is a doubly ruled surface containing two families of lines A and B . Two lines from the same set A or B do not meet, whereas any two lines chosen one from each set will always meet. Assuming that the lines λ_i lie on a quadric surface, since they do not meet, they must all come from the same family, which we assume to be A . Now consider any point x on the quadric surface. There is a unique line passing through x and belonging to the class B . This line must meet each of the lines λ_i , which belong to class A .

We are led therefore to consider the locus of all points x in \mathcal{P}^3 for which there exists a line passing through x meeting all the lines $\lambda_i, i = 1, \dots, 3$. To this end, let $x = (x, y, z, t)^T$ be a point on this locus. For each of the lines λ_i we may define a plane π_i passing

through x and λ_i . The condition that there exists a line passing through x meeting each λ_i means that the three planes π_i meet along that line.

Next, we formulate this last condition algebraically and give a method of computing the formula for the quadric surface. As before, letting $(a_{i1}, a_{i2}, a_{i3}, a_{i4})^T$ and $(b_{i1}, b_{i2}, b_{i3}, b_{i4})^T$ be two points on the line λ_i , the plane π_i passing through $x = (x, y, z, t)^T$ and the line λ_i may be computed as follows. Consider the matrix

$$\begin{pmatrix} a_{i1} & a_{i2} & a_{i3} & a_{i4} \\ b_{i1} & b_{i2} & b_{i3} & b_{i4} \\ x & y & z & t \end{pmatrix} \quad (6)$$

The plane π_i is given by the homogeneous vector $(p_{i1}, p_{i2}, p_{i3}, p_{i4})^T$ where $(-1)^j p_{ij}$ is the determinant of the 3×3 matrix obtained by deleting the j -th column of (6). Consequently, each p_{ij} is a homogeneous linear expression in x, y, z and t . Furthermore, since point $(x, y, z, t)^T$ lies on this plane it follows that

$$xp_{i1} + yp_{i2} + zp_{i3} + tp_{i4} = 0 \quad (7)$$

Now the fact that the three planes π_i meet along a common line translates into the algebraic fact that the rank of the matrix

$$P = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix}$$

is 2. This is equivalent to the condition

$$\det(P^{(j)}) = 0 \quad \text{for all } j, \quad (8)$$

where $P^{(j)}$ is the matrix obtained by removing the j -th column of P . Since each entry p_{ij} of P is a linear homogeneous expression in the variables x, y, z and t , the determinant $\det(P^{(j)})$ is a cubic homogeneous polynomial. A point on the required locus must satisfy the condition $\det(P^{(j)}) = 0$ for $j = 1, \dots, 4$. However, because of condition (7) these four equations are not independent. In particular, if p_j represents the j -th column of P , then (7) implies a relation

$$xp_1 + yp_2 + zp_3 + tp_4 = 0$$

Then

$$\begin{aligned} x \det(P^{(4)}) &= x \det(p_1 \ p_2 \ p_3) \\ &= \det(xp_1 \ p_2 \ p_3) \\ &= \det(-yp_2 - zp_3 - tp_4 \ p_2 \ p_3) \\ &= \det(-tp_4 \ p_2 \ p_3) \\ &= -t \det(p_2 \ p_3 \ p_4) \\ &= -t \det(P^{(1)}) \end{aligned} \quad (9)$$

This equation implies that x divides $\det(P^{(1)})$ and t divides $\det(P^{(4)})$. Furthermore, applying the same argument to other coordinates gives rise to an equation

$$\begin{aligned} \det(P^{(1)})/x &= -\det(P^{(2)})/y = \det(P^{(3)})/z = \\ &= -\det(P^{(4)})/t = R(x, y, z, t) \end{aligned}$$

where $R(x, y, z, t)$ is some homogeneous degree-2 polynomial. Then the defining equations (8) of the locus become

$$xR(x, y, z, t) = yR(x, y, z, t) = zR(x, y, z, t) = tR(x, y, z, t) = 0.$$

This implies that either $R(x, y, z, t) = 0$ or $x = y = z = t = 0$. The latter condition can be discounted, since $(0, 0, 0, 0)$ is not a valid set of homogeneous coordinates. Consequently, the desired locus is described by the degree-2 polynomial equation $R(x, y, z, t) = 0$, and is therefore a quadric surface. Since it is easily verified that the three original lines λ_i lie on this surface, the proof of the lemma is complete. \square

It is now a simple matter to prove the existence of transversals.

Theorem 2. *There exist exactly two transversals to four lines in general position in \mathcal{P}^3 .*

Proof. We choose three of the lines λ_1, λ_2 and λ_3 and construct the quadric surface S that they all line on. Let \mathbf{x}_1 and \mathbf{x}_2 be the two points of intersection of the fourth line λ_4 with the quadric surface. The construction of S in Lemma 1 shows that any transversal to lines λ_1, λ_2 and λ_3 must lie on S . Further, the lines λ_1, λ_2 and λ_3 all belong to one of the families, A , of ruled lines on the quadric surface, S . Let τ_1 and τ_2 be the lines in the other family B passing through \mathbf{x}_1 and \mathbf{x}_2 . Then τ_1 and τ_2 are the two transversals to all four lines. \square

Of course, it is possible that λ_4 does not meet the surface S in any real point, or is tangent to S . The statement of the theorem must be interpreted as allowing complex or double solutions. In the case of four real lines in space, there are either two real transversals or two conjugate complex transversals. In the case of complex transversals, there is no conceptual difficulty in defining the invariants as in the real case. The cross-ratio of points of intersections of the lines with the two conjugate transversals will result in two invariants which are complex conjugates of each other.

Various degenerate sets of lines also allow two transversals. For instance suppose that λ_1 and λ_2 are coincident, and so are λ_3 and λ_4 . One transversal to the four lines passes through the two points of intersection of the pairs of lines. The other transversal is the line of intersection of the two planes defined by λ_1, λ_2 and by λ_3, λ_4 . The cross-ratio invariant corresponding to the first transversal is zero, but the invariant corresponding to the second transversal is in general non-zero and is a useful invariant for this geometric configuration. This is similar to what happens for the algebraically defined invariants (see Section 2.1).

2.5 Independence and Completeness

I shall now show that the two geometrically defined invariants are independent and together completely characterize the set of four lines up to a projectivity of \mathcal{P}^3 .

To show independence, we start by selecting τ_1 and τ_2 , two arbitrary non-intersecting lines in space to

serve as transversals. Next, we mark off points a_1, a_2, a_3 and a_4 along τ_1 in such a way that their cross ratio is equal to any arbitrarily chosen invariant value. Similarly, mark off along τ_2 points b_1, b_2, b_3 and b_4 having another arbitrarily chosen cross-ratio invariant value. Now, joining a_i to b_i for each i gives a set of four lines having the two arbitrarily chosen invariants.

Next, it will be shown that the two invariants completely characterize the set of four lines up to a projectivity. Consequently, let four lines in space have two given cross-ratio invariant values with respect to transversals τ_1 and τ_2 respectively. Let the points of intersection of the four lines with τ_1 be a_1, a_2, a_3 and a_4 and the intersection points with τ_2 be b_1, b_2, b_3 and b_4 . Let a second set of lines with the same invariants be given, with transversals τ'_j and intersection points a'_i and b'_i . Our goal is to demonstrate that there is a projectivity taking τ_j to τ'_j for $j = 1, 2$, taking points a_i to a'_i and b_i to b'_i for $i = 1, \dots, 4$. It will follow that the projectivity takes one set of lines λ_i onto the other set.

Choosing two points on each of τ_1 and τ_2 , four points in all, and two points on each of τ'_1 and τ'_2 a further four points, there exists a projectivity taking the first set of four points to the second set, and hence taking τ_1 to τ'_1 and τ_2 to τ'_2 . Suppose that this projectivity takes a_i to a'_i and b_i to b'_i , it remains to be shown that there exists a projectivity preserving τ'_1 and τ'_2 and taking a'_i to a'_i and b'_i to b'_i . Without loss of generality it may be assumed that τ'_1 is the line $x = y = 0$ and that τ'_2 is the line $z = t = 0$. With this choice, we see that a projectivity of \mathcal{P}^3 represented by

a matrix of the form $\begin{pmatrix} H_1 & 0 \\ 0 & H_2 \end{pmatrix}$, where each H_j is

a 2×2 block, maps each τ'_j to itself. Furthermore each H_j represents a homography of the line τ'_j . Since the points a'_i and a''_i on τ'_1 have the same cross-ratio, there is a homography of τ'_1 taking a'_i to a''_i for $i = 1, \dots, 4$, and the same can be said for the points b'_i and b''_i on τ'_2 . Hence by independent choice of the two 2×2 matrices H_1 and H_2 , both mappings can be carried out simultaneously and the proof is complete.

2.6 Existence of an Isotropy

Four lines in \mathcal{P}^3 can be represented by a total of 16 independent parameters. On the other hand, there are 15 degrees of freedom for projectivities of \mathcal{P}^3 . This suggests that there should be only one invariant for four lines in space, but we have seen that there are two. The discrepancy arises because of the existence of an isotropy ([Mundy-92a]). To understand this, we need to determine the subgroup of all projectivities of \mathcal{P}^3 that fix four given lines. Any such projectivity will also fix the two transversals as well as the four points of intersection of the lines with each transversal. Since four points on each transversal are fixed, every point on the transversal must be fixed. This shows that a projectivity of \mathcal{P}^3 fixes four given lines if and only if it fixes the two transversals pointwise. Assuming as before that the two transversals are the lines $x = y = 0$ and $z = t = 0$, it is easily seen that a projectivity fixes the transversals pointwise if and only if it is rep-

represented by a matrix of the form $\text{diag}(k_1, k_1, k_2, k_2)$ where k_1 and k_2 are two independent constants. Allowing for an arbitrary scale factor in the matrix, this implies that there is a one-parameter subgroup of projectivities fixing the four lines. This reduces the number of degrees of freedom of the group action of projectivities of \mathcal{P}^3 on sets of four lines in space to 14, and explains why there are two independent invariants.

2.7 Relationship of Geometric to Algebraic Invariants

The fact that for real lines the algebraic invariants defined in Section 2.1 must be real whereas the geometric invariants may be complex indicates that they are not the same. However, since the geometric invariants completely determine the four lines up to projectivity, it must be possible to determine the algebraic invariants given the values of the geometric ones. Consider four lines with geometric invariants α and β . We desire to determine the values of the algebraic invariants given by (5). To this end, we may assume that the transversals are the lines $x = y = 0$ and $z = t = 0$ and that the points of intersections of the four lines with the transversals have coordinates

$$\begin{aligned} \mathbf{a}_2 &= (0, 0, 0, 1)^T \\ \mathbf{a}_1 &= (0, 0, \alpha, 1)^T \\ \mathbf{a}_3 &= (0, 0, 1, 1)^T \\ \mathbf{a}_4 &= (0, 0, 1, 0)^T \end{aligned}$$

and

$$\begin{aligned} \mathbf{b}_2 &= (0, 1, 0, 0)^T \\ \mathbf{b}_1 &= (\beta, 1, 0, 0)^T \\ \mathbf{b}_3 &= (1, 1, 0, 0)^T \\ \mathbf{b}_4 &= (1, 0, 0, 0)^T. \end{aligned}$$

These points have cross-ratio invariants α and β on the transversal lines $x = y = 0$ and $z = t = 0$ respectively.

From this it is easy to compute the value of the invariant (5) to be

$$I = (\alpha\beta, 1, 1 + \alpha\beta - \alpha - \beta). \quad (10)$$

Hence, it is easy to compute the algebraic invariants from the geometric ones. Similarly, given I , it is easy to solve (10) for α and β , which indicates that the algebraic invariant (5) is complete.

3 Computation of Line Invariants

It will be shown in this section that invariants of lines in space may be computed from two images with uncalibrated cameras, provided that the epipolar correspondence is known (in the sense to be explained below).

3.1 Camera Models

Nothing will be assumed about the calibration of the two cameras that create the two images. The camera model will be expressed in terms of a general projective transformation from three-dimensional real projective space, \mathcal{P}^3 , known as object space, to the two-dimensional real projective space \mathcal{P}^2 known as image space. The transformation may be expressed in

homogeneous coordinates by a 3×4 matrix P known as a camera matrix and the correspondence between points in object space and image space is given by $\mathbf{u}_i \approx P\mathbf{x}_i$, where the symbol \approx means equal up to multiplication by a non-zero scalar factor.

For convenience it will be assumed that the camera placements are not at infinity, that is, that the projections are not parallel projections. In this case, a camera matrix may be written in the form

$$P = (M \mid -Mt)$$

where M is a 3×3 non-singular matrix and \mathbf{t} is a column vector $\mathbf{t} = (t_x, t_y, t_z)^T$ representing the location of the camera in object space.

3.2 The Essential Matrix

Consider a set of points $\{\mathbf{x}_i\}$ as seen in two images. The set of points $\{\mathbf{x}_i\}$ will be visible at image locations $\{\mathbf{u}_i\}$ and $\{\mathbf{u}'_i\}$ in the two images. In normal circumstances, the correspondence $\{\mathbf{u}_i\} \leftrightarrow \{\mathbf{u}'_i\}$ will be known, but the location of the original points $\{\mathbf{x}_i\}$ will be unknown. As shown in [Higgins-81] there exists a matrix Q , called the essential matrix, such that

$$\mathbf{u}'_i^T Q \mathbf{u}_i = 0 \quad \text{for all } i. \quad (11)$$

Given at least 8 point correspondences, the matrix Q may be computed from (11). Longuet-Higgins ([Higgins-81]) suggested a linear solution of the equations (11). Other methods ([Horn-91, Tsai-84, Spetsakis-92]) have been suggested relying on properties of the essential matrix.

Although the essential matrix was originally defined for calibrated cameras, it may also be defined for uncalibrated cameras using the same equation (11). Methods of computing the essential matrix for uncalibrated cameras have been suggested using point correspondences ([Faugeras-92]) or line-correspondences ([Hartley-93b]).

For calibrated cameras, the essential matrix determines the camera matrices uniquely, up to a scaled Euclidean transformation¹. For uncalibrated cameras, this is not the case. The connection between essential matrix and camera matrices for uncalibrated cameras will be explained below. For proofs of the following theorems, see [Hartley-Gupta-92].

Given a vector, $\mathbf{t} = (t_x, t_y, t_z)^T$ it is convenient to introduce the skew-symmetric matrix

$$[\mathbf{t}]_{\times} = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix}. \quad (12)$$

Theorem 3. *If Q is an essential matrix corresponding to a pair of uncalibrated cameras, then Q factors as a product $Q = P[\mathbf{t}]_{\times}$ for some vector \mathbf{t} and non-singular matrix P . Then, one possible choice of camera matrices consistent with Q is given by*

$$M = (I \mid 0) \quad , \quad M' = (P^* \mid -P^* \mathbf{t})$$

where P^* is the inverse transpose of P .

¹Strictly speaking there are four possible solutions

Given a pair of camera matrices and some image correspondences $u_i \leftrightarrow u'_i$ it is easy to compute the corresponding object points x_i by the solution of a set of linear equations (in effect by triangulation). The pair of camera matrices given in Theorem 3 is not necessarily the correct pair, and hence the reconstructed set of object points will not necessarily be correct. However, the following theorem shows that the points are nevertheless correct up to a projectivity of \mathcal{P}^3 .

Theorem 4. Suppose Q is an essential matrix and M and M' are any pair of camera matrices consistent with Q . Let $u_i \leftrightarrow u'_i$ be corresponding points in the two images and $\{x_i\}$ be a set of object points such that $u_i \approx Mx_i$ and $u'_i \approx M'x_i$. Now let \tilde{M} and \tilde{M}' be a different pair of camera matrices consistent with Q and let $\{\tilde{x}_i\}$ be the respective set of object points. Then there is a projectivity h of \mathcal{P}^3 taking each x_i to \tilde{x}_i .

The algorithm for computing invariants may now be formulated in broad terms as follows.

1. Compute the essential matrix from the correspondences using any available algorithm.
2. Select a pair of camera matrices M and M' according to Theorem 3.
3. Reconstruct the scene geometry using the chosen camera matrices.
4. Compute invariants of the scene.

3.3 Computing Lines in Space

To be able to compute invariants of lines in space, it is sufficient to be able to compute the locations of the lines in \mathcal{P}^3 from their images in two views (step 3 of the above algorithm outline).

Lines in the image plane are represented as 3-vectors. For instance, a vector $l = (l, m, n)^T$ represents the line in the plane given by the equation $lu + mv + nw = 0$. Similarly, planes in 3-dimensional space are represented in homogeneous coordinates as a 4-dimensional vector $\pi = (p, q, r, s)^T$.

The relationship between lines in the image space and the corresponding plane in object space is given by the following lemma.

Lemma 5. Let λ be a line in \mathcal{P}^3 and let the image of λ as taken by a camera with transformation matrix M be l . The locus of points in \mathcal{P}^3 that are mapped onto the image line l is a plane, π , passing through the camera centre and containing the line λ . It is given by the formula $\pi = M^T l$.

Proof. A point x lies on π if and only if it is mapped to a point on the line l by the action of the transformation matrix. This means that Mx lies on the line l , and so

$$l^T Mx = 0. \quad (13)$$

On the other hand, a point x lies on the plane π if and only if $\pi^T x = 0$. Comparing this with (13) lead to the conclusion that $\pi^T = l^T M$ or $\pi = M^T l$ as required. \square

Now, given two images I and I' of a line λ in space as taken by two cameras with camera matrices M and M' , the line λ is the intersection of the planes $M^T l$ and $M'^T l'$.

4 Experimental Results

Three images of a pair of wooden blocks representing houses were acquired and vertices and edges were extracted. The images are shown in Figures 1, 2, and 3. Corresponding edges and vertices were selected by hand from among those detected automatically. The edges and vertices shown in Fig. 4 were chosen. There were 13 edges and 15 lines extracted from each of the images. The dotted edges were not visible in all images and were not chosen. Vertices are represented by numbers and edges by letters in the figure. Because of the way edges and vertices were found by the segmentation algorithm, the edges do not always pass precisely through the indicated vertices, but sometimes through a closely neighboring vertex. On other occasions, the full edge was not detected as a single edge, but was broken into several pieces. This is usual with most edge detection algorithms, and is a source of error in the computation of invariants.

The essential matrices Q_{12} for the first and second images and Q_{23} for the second and third images were computed from the point matches. Compatible set of camera matrices were computed, the locations of the lines in \mathcal{P}^3 were reconstructed and invariants (5) were computed algebraically.

4.1 Comparison of Invariant Values

The invariant (5) is represented as homogeneous vectors. Two such vectors are considered equivalent if they differ by a non-zero scale factor. Because of arithmetic error and image noise, two computed invariant values will rarely be exactly proportional. In order to compare two such computed invariant values (perhaps when attempting to match an object with a reference object), each homogeneous vector is multiplied by a scale factor chosen to normalize its length to 1. This normalization determines the vector up to a multiplication by a factor ± 1 . Two such normalized homogeneous vector invariants v_1 and v_2 are deemed close if v_1 is close to v_2 or to $-v_2$ using a Euclidean norm. Correspondingly, a metric may be defined by

$$d(v_1, v_2) = \left(1 - \left| \frac{v_1 \cdot v_2}{\|v_1\| \|v_2\|} \right| \right)^{1/2}. \quad (14)$$

For any v_1 and v_2 , distance $d(v_1, v_2)$ lies between 0 and 1. A value close to 0 means a very good match, whereas values close to 1 are mismatches.

4.2 Invariants of 4 lines

Six sets of four lines were chosen as in the following table, which shows the labels of the lines as given in Fig. 4).

$$\begin{aligned} S_1 &= \{B, C, J, K\} \\ S_2 &= \{B, G, J, N\} \\ S_3 &= \{A, B, H, I\} \\ S_4 &= \{B, D, E, G\} \\ S_5 &= \{A, C, O, J\} \\ S_6 &= \{B, I, L, N\} \end{aligned}$$

Table 15 shows the results. The (i, j) -th entry of the table shows the distance according to the metric (14) between the invariant of set S_i as computed from the first image pair with that of set S_j as computed from the second image pair. The diagonal entries of the matrix (in bold) should be close to 0.0, which indicates that the invariants had the same value when computed from different pairs of views.

The only bad entry in this matrix is in the position (4, 4). This is because of the fact that the four lines chosen contained three coplanar lines (lines B , D and E). This causes the values of the invariant to be indeterminate (that is $(0, 0, 0)$), and shows that such instances must be detected and avoided. The entry in position (3, 3) also shows instability for similar reasons.

0.013	0.674	0.303	0.689	0.643	0.449
0.647	0.034	0.741	0.838	0.707	0.222
0.062	0.691	0.229	0.708	0.708	0.461
0.287	0.608	0.182	0.890	0.856	0.384
0.657	0.722	0.900	0.719	0.003	0.694
0.473	0.239	0.555	0.948	0.719	0.033

(15)

One concludes from this experiment that the four-line invariant is a powerful discriminator between sets of four lines, but care must be taken to detect and exclude degenerate and near-degenerate cases.

Acknowledgement I am indebted to Joe Mundy for introducing me to the subject of projective invariants, and for many enlightening conversations during the preparation of this paper.

References

- [Abhyankar-92] S. S. Abhyankar, "Invariant Theory and Enumerative Combinatorics of Young Tableaux", In [Mundy-Zisserman-92], pp. 53-90, (1992).
- [Burns-92] J. B. Burns, R. S. Weiss and E. M. Riesenman, "The Non-existence of General-case View-invariants", In [Mundy-Zisserman-92], pp. 143-156, (1992).
- [Coelho-92] C. Coelho, A. Heller, J. Mundy, D. Forsyth and A. Zisserman, "An Experimental Evaluation of Projective Invariants", In [Mundy-Zisserman-92], pp. 103-124, (1992).
- [Faugeras-92] Faugeras, O., "What can be seen in three dimensions with an uncalibrated stereo rig?", Proc. of ECCV-92, G. Sandini Ed., LNCS-Series Vol. 588, Springer-Verlag, 1992, pp. 563 - 578.
- [Hartley-Gupta-92] R. Hartley, R. Gupta and Tom Chang, "Stereo from Uncalibrated Cameras" Proceedings of CVPR92.
- [Hartley-93a] R. Hartley, "Chirality Invariants", in these proceedings.
- [Hartley-93b] R. Hartley, "Camera Calibration Using Line Correspondences", in these proceedings.
- [Higgins-81] H.C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," Nature, Vol. 293, 10 Sept. 1981.
- [Horn-91] B.K.P. Horn, "Relative Orientation Revisited", Journal of the Optical Society of America, A Vol 8, number 10, pp. 1630-1638, Oct 91.
- [Mundy-Zisserman-92] J. L. Mundy and A. Zisserman (editors), "Geometric Invariance in Computer Vision," MIT Press, Cambridge Ma, 1992.
- [Mundy-92a] J. L. Mundy and A. Zisserman. "Introduction - towards a new framework for vision" In [Mundy-Zisserman-92], pp. 1-49.
- [Semple-Kneebone-52] J.G. Semple and G. T. Kneebone "Algebraic Projective Geometry" Oxford University Press, (1952), ISBN 0 19 8531729.
- [Spetsakis-92] Minas E. Spetsakis and Yiannis Aloimonos, "Optimal Visual Motion Estimation," IEEE Trans. Patt. Anal. Machine Intell., Vol PAMI-14, No. 9, September 1992, pp 959 - 964.
- [Tsai-84] R. Y. Tsai and T. S. Huang, "Uniqueness and estimation of three dimensional motion parameters of rigid objects with curved surfaces", IEEE Trans. Patt. Anal. Machine Intell., vol. PAMI-6, pp. 13-27, 1984.
- [Weng-92] J. Weng, T.S. Huang, and N. Ahuja, "Motion and Structure from Line Correspondences: Closed-Form Solution, Uniqueness and Optimization", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 14, No. 3, March, 1992.
- [Zisserman-92] A. Zisserman, D. A. Forsyth, J. L. Mundy and C. A. Rothwell, "Recognizing General Curved Objects Efficiently", In [Mundy-Zisserman-92], pp. 265-290, (1992).
- [Zisserman-Hartley-93] A. Zisserman, R. Hartley, J. Mundy, P. Beardsley, "Projective Structure From Multiple Views", in preparation.

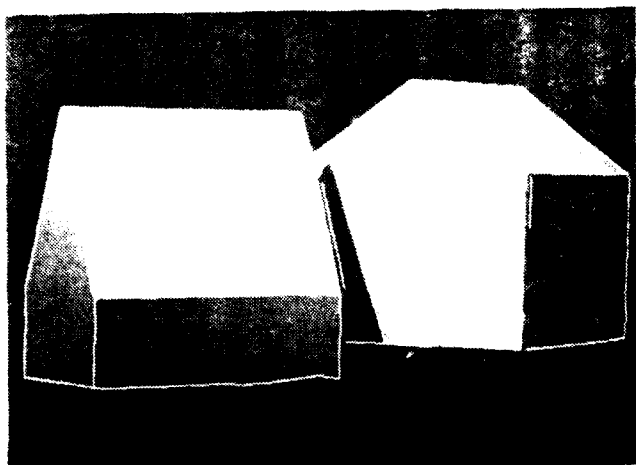


Figure 1. First view of houses

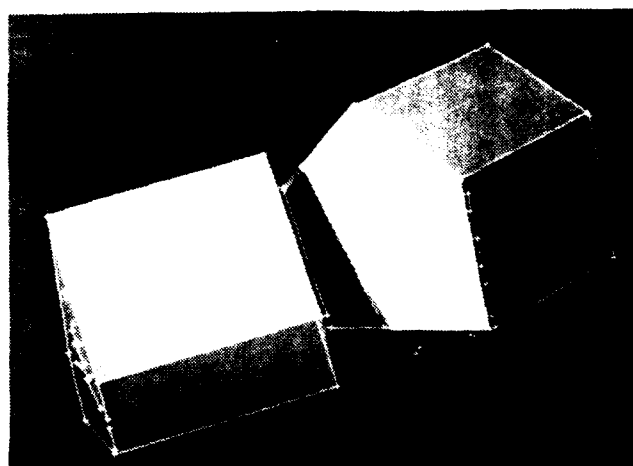


Figure 3. Third view of houses

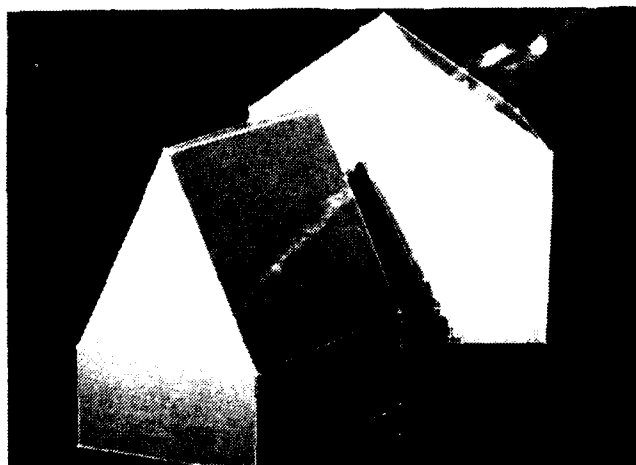


Figure 2. Second view of houses

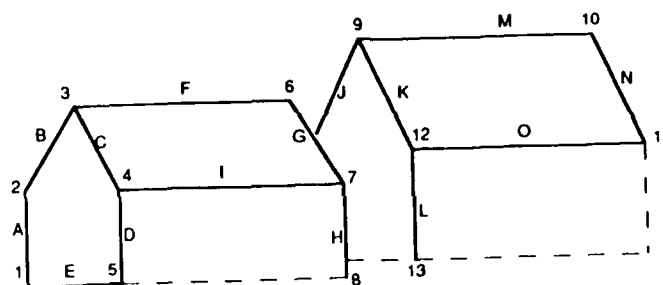


Figure 4. Selected vertices and edges

Figure 4. Selected vertices and edges

Cheirality Invariants

Richard I. Hartley

GE - Corporate Research and Development,
P.O. Box 8, Schenectady, NY, 12301.

Ph : (518)-387-7333

Fax : (518)-387-6845

email : hartley@crd.ge.com

Abstract

It is known that a set of points in 3 dimensions is determined up to projectivity from two views with uncalibrated cameras. It is shown in this paper that this result may be improved by distinguishing between points in front of and behind the camera. Any point that lies in an image must lie in front of the camera producing that image. Using this idea, it is shown that the scene is determined from two views up to a more restricted class of mappings known as good projectivities, which are precisely those projectivities that preserve the convex hull of an object of interest. An invariant of good projectivity known as the cheirality invariant of a set of points is defined and it is shown how the cheirality invariant may be computed using two uncalibrated views. As demonstrated theoretically and by experiment the cheirality invariant may distinguish between sets of points that are projectively equivalent (but not via a good projectivity). These results lead to necessary and sufficient conditions for a set of corresponding pixels in two images to be realizable as the images of a set of points in 3 dimensions.

Using similar methods, a necessary and sufficient condition is given for the orientation of a set of points to be determined by two views. If the perspective centres are not separated from the point set by a plane, then the orientation of the set of points is determined from two views.

Good projectivities and the cheirality invariant are also defined for point sets in a plane, which allows these new methods to be applied to images of planar objects.

1 Introduction

Consider a set of points $\{x_i\}$ lying in a plane in space and let $\{u_i\}$ and $\{u'_i\}$ be two images of these points taken with arbitrary uncalibrated perspective (pinhole) cameras. It is well known that the points u_i and u'_i are related by a planar projectivity, that is, there exists h a projectivity of the plane such that $hu_i = u'_i$ for all i . This fact has been used for the recognition of planar objects. For instance in [Rothwell-92] planar projective invariants were used to define indexing functions allowing look-up of the objects in an object data-base. Since the indexing functions are invariant under plane projectivities, they

provide the same value independent of the view of the object.

In a similar way, it has been shown in [Faugeras-92] and [Hartley-Gupta-92] that a set of points in 3-dimensions is determined up to a 3-dimensional projectivity by two images taken with uncalibrated cameras. Both these papers give a constructive method for determining the point configuration (up to projectivity). This permits the computation of projective invariants of sets of points seen in two views. An experimental verification of these results has been reported in [Hartley-92] and is summarized in this paper.

The papers just cited make no distinction between points that lie in front of the camera and those that lie behind. The specification of the front of a camera will be termed the *cheirality* of the camera (from Greek : $\chi\epsilon\iota\rho$ = *hand* or *side*). It is well known that camera cheirality is valuable in determining scene geometry for calibrated cameras. Longuet-Higgins [Higgins-81] uses it to distinguish between four different scene reconstructions. No systematic treatment of cheirality of uncalibrated cameras has previously appeared, however. Investigation of this phenomenon turns out to be quite fruitful, as is seen in the present paper. Cheirality is valuable in distinguishing different point sets in space, especially in allowing projectively equivalent point sets to be distinguished.

The major results of this paper are summarized now. In Definition 4 a class of projectivities called *good projectivities* is defined, consisting of those ones that preserve the convex hull of a set of points of interest. In section 5 an invariant of good projectivity is defined - the cheirality invariant. Theorem 13 strengthens the result of [Faugeras-92, Hartley-Gupta-92] by showing that a 3-dimensional point set is determined up to good projectivity by its image in two uncalibrated views. This sharpening of the theorem of [Faugeras-92, Hartley-Gupta-92] results from a consideration of the cheirality of the cameras. In section 9 an example of computation of the cheirality invariant for 3D point sets shows that it is useful in distinguishing between non-equivalent point sets. In section 7 the concept of good-projectivity is applied to orientation of point sets, explaining why some point sets allow two differently oriented reconstructions from two views, whereas some do not. The relationship of this result to human visual perception of 3D scenes is

briefly mentioned, suggesting that the brain accepts various interpretations of a scene differing by good projectivities, but not by arbitrary projectivities.

2 Notation

We will consider object space to be the 3-dimensional Euclidean space R^3 and represent points in R^3 as 3-vectors. Similarly, image space is the 2-dimensional Euclidean space R^2 and points are represented as 2-vectors. Euclidean space, R^3 is embedded in a natural way in projective 3-space P^3 by the addition of a plane at infinity. Similarly, R^2 may be embedded in the projective 2-space P^2 by the addition of a line at infinity. The simplicity of considering projections between P^3 and P^2 has led many authors to identify P^3 and P^2 as the object and images space. This point of view will not be followed here however, although when necessary we will consider points in R^2 and R^3 to as lying in P^2 and P^3 respectively, via the natural embedding.

Vectors will be represented as bold-face lower case letters, such as \mathbf{x} . Such a notation represents a column vector. The corresponding row vector will be denoted by \mathbf{x}^T . The notation \mathbf{x} usually denotes a vector in R^3 , whereas \mathbf{u} represents a vector in R^2 . Elements in projective spaces P^3 and P^2 will be denoted with a tilde accent. For instance, $\tilde{\mathbf{x}}$ is a homogeneous 4-vector representing an element in P^3 , and $\tilde{\mathbf{u}}$ is a homogeneous 3-vector representing an element of P^2 .

The notation \approx represents equality of matrices or homogeneous vectors up to an arbitrary non-zero factor. If $\mathbf{x} = (x, y, z)^T$ is a 3-vector representing a point in R^3 , then $\tilde{\mathbf{x}}$ is the vector $(x, y, z, 1)^T$. Similarly, if $\mathbf{u} = (u, v)^T$, then $\tilde{\mathbf{u}}$ represents the vector $(u, v, 1)^T$.

The notation $a \doteq b$ means that a and b have the same sign. For instance $a \doteq 1$ has the same meaning as $a > 0$.

3 Projections in P^3

A projection from P^3 into P^2 is represented by a 3×4 matrix P , whereby a point $\tilde{\mathbf{x}}$ maps to the point $\tilde{\mathbf{u}} \approx P\tilde{\mathbf{x}}$. It will be assumed that P has rank 3. Since P has 4 columns but rank 3, there is a unique point $\tilde{\mathbf{t}}$ such that $P\tilde{\mathbf{t}} = (0, 0, 0)^T$. In other words, the projective transformation is undefined at the point $\tilde{\mathbf{t}}$, since $(0, 0, 0)^T$ is not a valid homogeneous 3-vector. The point $\tilde{\mathbf{t}}$ will be called the *perspective centre* of the camera. We will assume that the perspective centre is not a point at infinity so we may write $\tilde{\mathbf{t}} \approx \tilde{\mathbf{t}} = \begin{pmatrix} t \\ 1 \end{pmatrix}$

where \mathbf{t} is the perspective center as a point in R^3 .

Now, the camera matrix P may be written in block form as $P = (M | \mathbf{c})$ where M is a 3×3 block and \mathbf{c} is a column vector. Now

$$P\tilde{\mathbf{t}} = (M | \mathbf{c}) \begin{pmatrix} \mathbf{t} \\ 1 \end{pmatrix} = M\mathbf{t} + \mathbf{c} = 0,$$

and so $\mathbf{c} = -M\mathbf{t}$. In future, we will write $P = (M | -M\mathbf{t})$. Now since P has rank 3 and $-M\mathbf{t}$ is a linear

combination of the columns of M , it follows that M must have rank 3. In other words, M is non-singular. Summarizing this discussion we have

Proposition 1. *If P is a camera transform matrix for a camera with perspective centre not at infinity, then P can be written as $P = (M | -M\mathbf{t})$ where M is a non-singular 3×3 matrix and \mathbf{t} represents the perspective centre in R^3 .*

There exist points in P^3 that are mapped to points at infinity in the image. To find what they are, we suppose that $\tilde{\mathbf{u}} = (u, v, 0)^T = P\tilde{\mathbf{x}}$. Letting \mathbf{p}_1^T , \mathbf{p}_2^T and \mathbf{p}_3^T be the rows of P , we see that $\mathbf{p}_3^T \tilde{\mathbf{x}} = 0$. In other words, a point $\tilde{\mathbf{x}}$ in P^3 that maps to a point at infinity in the image must satisfy the equation $\tilde{\mathbf{x}}^T \mathbf{p}_3 = 0$. Looked at another way, if \mathbf{p}_3 is taken as representing a plane in P^3 , then a point $\tilde{\mathbf{x}}$ lies on the plane \mathbf{p}_3 if and only if $\tilde{\mathbf{x}}^T \mathbf{p}_3 = 0$. In other words, the condition for $\tilde{\mathbf{x}}$ to map to a point at infinity is the same as the condition for $\tilde{\mathbf{x}}$ to lie on the plane \mathbf{p}_3 . Since $P\tilde{\mathbf{t}} = 0$, we see in particular that $\mathbf{p}_3^T \tilde{\mathbf{t}} = 0$, and so $\tilde{\mathbf{t}}$ lies on the plane \mathbf{p}_3 . To summarize this paragraph, the set of points in P^3 mapping to a point at infinity in the image is a plane passing through the perspective centre and represented by \mathbf{p}_3 , where \mathbf{p}_3^T is the last row of P . This plane will be called the *meridian plane* of the camera.

Restricting now to R^3 , consider a point \mathbf{x} in space, not lying on the meridian plane. It is projected by the camera with matrix P onto a point \mathbf{u} where $w\mathbf{u} = P\tilde{\mathbf{x}}$ for some scale factor w . The value of w will vary continuously with \mathbf{x} and the set of points where it vanishes is precisely the meridian plane. It follows that on one side of the meridian plane $w > 0$ and on the other side, $w < 0$. It can be shown, but is not used in this paper, that w is in fact proportional to the distance of \mathbf{x} from the meridian plane.

Any real camera can only view points on one side of the meridian plane, those points that are "in front of" the camera. Points on the other side will not be visible. In order to distinguish the front of the camera from the back, a convention is necessary.

Definition 2. A camera matrix $P = (M | -M\mathbf{t})$ is said to be *normalized* if $\det(M) > 0$. If P is a normalized camera matrix, a point \mathbf{x} is said to lie *in front* of the camera if $P\tilde{\mathbf{x}} = w\tilde{\mathbf{u}}$ with $w > 0$. Points \mathbf{x} for which $w < 0$ are said to be *behind* the camera.

Any camera matrix may be normalized by multiplying it by -1 if necessary. It will always be assumed that camera matrices are normalized. The selection of which side of the camera is the front is simply a convention, consistent with the assumption that for a camera with matrix $(I | 0)$, points with positive z -coordinate lie in front of the camera. This is the usual convention in computer vision literature, used for instance in [Higgins-81].

The following statement expresses the fact that a camera sees only those points that lie in front of it.

Proposition 3. *A point \mathbf{x} in R^3 is mapped to a point \mathbf{u} in R^2 by a camera with normalized matrix P if and only if $w\mathbf{u} = P\tilde{\mathbf{x}}$ for some constant $w > 0$.*

4 Good Projectivities

A subset B of R^n is called convex if the line segment joining any two points in B also lies entirely within B . The convex hull of B , denoted \bar{B} is the smallest convex set containing B .

Definition 4. Let B be a subset of R^n and let h be a projectivity of P^n . The projectivity h is said to be a "good projectivity" with respect to the set B if $h^{-1}(L_\infty)$ does not meet \bar{B} , where L_∞ is the plane (or line) at infinity.

A good projectivity with respect to B is precisely one that preserves the convex hull of B . It may be verified that if h is a good projectivity with respect to B , then h^{-1} is a good projectivity with respect to $h(B)$. Details are omitted for the sake of brevity. We will be considering sets of points $\{u_i\}$ and $\{u'_i\}$ that correspond via a projectivity. When we speak of the projectivity being *good*, we will mean good with respect to the set $\{u_i\}$.

An alternative characterization of good projectivities is given in the following theorem.

Theorem 5. A projectivity $h : P^n \rightarrow P^n$ represented by a matrix H is good with respect to a set $B = \{u_i\} \subset R^n - h^{-1}(L_\infty)$ if and only if there exist constants w_i , all of the same sign, such that $H\bar{u}_i = w_i\bar{u}'_i$.

Proof. To prove the forward implication, we assume that h is a good projectivity. By definition, constants w_i exist such that $H\bar{u}_i = w_i\bar{u}'_i$. What needs proof is that they all have the same sign. The value of w in the mapping $w\bar{u}'_i = H\bar{u}_i$ is a continuous function of the point u . If $w_i > 0$ for some point u_i , and $w_j < 0$ for another point u_j , then there must be some point u_∞ on the line segment joining u_i to u_j for which $w = 0$. This means that $h(u_\infty)$ lies on the line at infinity, contrary to hypothesis.

Next, to prove the converse, we assume that there exist such constants w_i all of the same sign. Let S be the subset of R^n consisting of all points u satisfying the condition $H\bar{u} = w\bar{u}'$ such that w has the same sign as all w_i . The set S contains B and it is clear that $S \cap h^{-1}(L_\infty) = \emptyset$. All that remains to show is that S is convex, for then S must contain the convex hull of B . If u_i and u_j are points in S with corresponding constants w_i and w_j , then any intermediate point u between u_i and u_j must have w value intermediate between w_i and w_j . Consequently, the value of w must have the same sign as w_i and w_j , and so u lies in S also. This completes the proof. \square

As just noted, if a projectivity is not good, then there are points in the convex hull for which w equals nought (0). For this reason, a projectivity that is not good will be called "naughty".¹

This theorem gives an effective method of identifying good projectivities. The question remains whether good projectivities form a useful class. This question will be answered by the following theorem.

¹This terminology was suggested to me by David Forsyth

Theorem 6. If B is a point set in a plane (the "object plane") in R^3 lying entirely in front of a projective camera, then the mapping from the object plane to the image plane defined by the camera is a good projectivity with respect to B .

Proof. That there is a projectivity h mapping the object plane to the image plane is well known. What is to be proven is that the projectivity is good with respect to B . Let L be the line in which the meridian plane of the camera meets the object plane. Since B lies entirely in front of the camera, L does not meet the convex hull of B . However, by definition of the meridian plane $L = h^{-1}(L_\infty)$, where L_∞ is the line at infinity in the image plane. Therefore, h is a good projectivity with respect to B . \square

As an example, Fig. 1 shows an image of a comb and the image resampled according to a naughty projectivity. Most people will agree that the resampled image is unlike any view of a comb seen by camera or human eye. Nevertheless, the two images are projectively equivalent and will have the same projective invariants.

Note that if points u_i are visible in an image, then the corresponding object points must lie in front of the camera. Applying Theorem 6 to a sequence of imaging operations (for instance, a picture of a picture of a picture, etc), it follows that the original and final images in the sequence are connected by a planar projectivity which is good with respect to any point set in the object plane visible in the final image.

Similarly, if two images are taken of a set of point $\{x_i\}$ in a plane, u_i and u'_i being corresponding points in the two images, then there is a good projectivity (with respect to the u_i) mapping each u_i to u'_i , and so Theorem 5 applies, yielding the following proposition.

Proposition 7. If $\{u_i\}$ and $\{u'_i\}$ are corresponding points in two views of a set of object points $\{x_i\}$ lying in a plane, then there is a matrix H representing a planar projectivity such that $H\bar{u}_i = w_i\bar{u}'_i$ and all w_i have the same sign.

This fact was pointed out to me by Charles Rothwell (private communication) and served as a starting point for the current investigation. Rothwell derived this result using the methods of [Sparr-92].

5 An integer valued invariant

Given a set of $N \geq n+2$ points $\{u_i\}$, $i = 1, \dots, N$ in R^n , it is possible to define an invariant of good projectivity as follows. Let e_1, \dots, e_{n+2} be points in R^n such that $\{\bar{e}_i\}$ form a canonical projective basis for P^n . For $n = 2$, the points $(0,0)^T$, $(1,0)^T$, $(0,1)^T$ and $(1,1)^T$ will do. Assume that the points u_i are numbered in such a way that the first $n+2$ of them are in general position (meaning that no $n+1$ of them lie in a codimension 1 hyperplane). In this case, there exists a projectivity g (not necessarily good) such that $g(u_i) = e_i$ for $i = 1, \dots, n+2$. Now, for each $i = 1, \dots, N$ we define a value η_i as follows. If $g(u_i)$ lies on the plane at infinity, we set $\eta_i = 0$. Otherwise, there exists a further e_i such that $g(u_i) = e_i$. If g is represented by a matrix

G , then η_i is defined by the equation $G\hat{u}_i = \eta_i \hat{e}_i$. We show that, except for possible simultaneous negation, the values $\text{sign}(\eta_i)$ are an invariant of good projectivity. Here $\text{sign}(\eta_i)$ is defined to equal 1, -1 or 0 depending on whether η_i is positive, negative or zero respectively. The invariant value is of course dependent on the choice of canonical basis $\{\hat{e}_i\}$.

To prove the invariance, suppose that h is a good projectivity with respect to points $\{u_i\}$ and let $h(u_i) = u'_i$. Consider the projectivity g' defined by $g'(u'_i) = e_i$ for $i = 1, \dots, n+2$. Values η'_i may be defined as before in terms of the projectivity g' . On the other hand, values w_i may be defined in terms of the projectivity h mapping each u_i to u'_i as in Theorem 5.

Since h and $g'^{-1} \circ g$ agree on a set of basis points, it follows that $h = g'^{-1} \circ g$. Consequently, $w_i = \eta_i / \eta'_i$. However, under the assumption that h is a good projectivity, all the w_i have the same sign, and so, for all i , we have $\eta_i = \epsilon \eta'_i$, where $\epsilon = \pm 1$. In other words, the set of values $\text{sign}(\eta_i)$ are an invariant under good projectivity, except for possible simultaneous negation.

It is possible to code the values η_i into a single number according to the formula

$$\chi(u_1, u_2, \dots, u_N) = \left| \sum_{i=1}^N \text{sign}(\eta_i) 3^{i-1} \right| \quad (1)$$

The value $\chi(u_i)$ is an invariant under good projectivity of the ordered set of points u_i . It will be called the *chirality invariant* of the points.

6 Three dimensional point sets

We now consider three-dimensional point sets. The question that will be addressed is: "Under what conditions can points u_i and u'_i in two views be the images of a three dimensional point set x_i corresponding to two arbitrary uncalibrated cameras?". One well-known necessary condition ([Higgins-81]) is the epipolar constraint, $\hat{u}_i^T Q \hat{u}_i = 0$ for all i and some rank-two matrix Q . We will ignore the effects of noise, so that the epipolar constraint equation will be assumed to hold exactly. The question is whether this is also a sufficient condition. The answer is no.

It will be assumed that there are sufficient points for the matrix Q to be determined unambiguously, that is at least 7 ([Hartley-92]) or 8 ([Higgins-81]) points. Under these conditions as shown in [Hartley-Gupta-92] and [Faugeras-92] it is possible to determine the location of points \hat{x}_i and cameras P and P' such that $\hat{u}_i \approx P\hat{x}_i$ and $\hat{u}'_i \approx P'\hat{x}_i$, and furthermore, the choice is unique up to projectivity of P^3 . Assuming that none of the reconstructed points x_i is at infinity, we can write

$$\begin{aligned} w_i \hat{u}_i &= P\hat{x}_i \\ w'_i \hat{u}'_i &= P'\hat{x}_i \end{aligned} \quad (2)$$

If all the w_i and w'_i are positive, then according to Proposition 7 the points x_i map to points u_i and u'_i in the two images. Normally, this will not be the case. It is possible, however, that another choice of P , P' and x_i exists with the desired property.

We introduce some new terminology. A triplet $(Q, \{u_i\}, \{u'_i\})$ is called an *epipolar configuration* if Q is a rank 2 matrix satisfying the epipolar constraint equation $\hat{u}_i^T Q \hat{u}_i = 0$ for all i . A *weak realization* of $(Q, \{u_i\}, \{u'_i\})$ is a triplet $(P, P', \{x_i\})$, where P and P' are a choice of normalized camera matrices corresponding to the essential matrix Q and the points $\{x_i\}$ are object points satisfying the equations (2) for each i . A *strong realization* is such a triplet satisfying the additional condition that all the w_i and w'_i are positive. The triplet $(Q, \{u_i\}, \{u'_i\})$ is called a *feasible configuration* if a strong realization exists.

The following lemma sets notation and derives a basic technical result.

Lemma 8. Let $(P, P', \{x_i\})$ and $(\bar{P}, \bar{P}', \{\bar{x}_i\})$ be two weak realization of a feasible configuration $(Q, \{u_i\}, \{u'_i\})$. There exists a 4×4 matrix H such that $P \approx \bar{P}H$, $P' \approx \bar{P}'H$ and $\hat{x}_i \approx H^{-1}\hat{\bar{x}}_i$. Assume that P , P' , \bar{P} and \bar{P}' are normalized and let constants ϵ , η_i , w_i and \bar{w}_i be defined by the equations

$$\begin{aligned} P &= \epsilon \bar{P}H \\ \hat{x}_i &= \eta_i H^{-1} \hat{\bar{x}}_i \\ w_i \hat{u}_i &= P\hat{x}_i \\ \bar{w}_i \hat{\bar{u}}_i &= \bar{P}\hat{\bar{x}}_i \end{aligned} \quad (3)$$

Then $w_i \bar{w}_i \epsilon \eta_i = 1$.

If constants w'_i , \bar{w}'_i , and ϵ' are defined in a similar way then $w'_i \bar{w}'_i \epsilon' \eta_i = 1$.

Proof. The existence of the matrix H is proven in [Hartley-Gupta-92]. Now,

$$\begin{aligned} w_i \hat{u}_i &= P\hat{x}_i \\ &= \epsilon \eta_i \bar{P}H H^{-1} \hat{\bar{x}}_i \\ &= \epsilon \eta_i \bar{w}_i \hat{\bar{u}}_i \end{aligned}$$

whence $w_i = \epsilon \eta_i \bar{w}_i$. Multiplying each side of this equation by w_i gives the required result. The proof for the primed quantities is of course the same. \square

A further useful technical result follows.

Lemma 9. Let H be the matrix

$$H = \begin{pmatrix} I & 0 \\ k\mathbf{v}^T & k \end{pmatrix}.$$

Then with the notation used in Lemma 8, $\epsilon = \hat{\mathbf{v}}^T \hat{\mathbf{t}}$, $\epsilon' = \hat{\mathbf{v}}^T \hat{\mathbf{t}}'$ and for each i , $\eta_i = k\hat{\mathbf{v}}^T \hat{\mathbf{x}}_i$, where \mathbf{t} and \mathbf{t}' are the perspective centres of P and P' . (Remember that \doteq denotes equality of sign.)

Proof. One verifies that

$$H^{-1} = \begin{pmatrix} I & 0 \\ -\mathbf{v}^T & k^{-1} \end{pmatrix}.$$

Let $P = (M \mid -M\mathbf{t})$ with $\det(M) > 0$ and $\bar{P} = (\bar{M} \mid -\bar{M}\mathbf{t})$ with $\det(\bar{M}) > 0$. Then from the $\epsilon \bar{P} = PH^{-1}$ it

follows that $\epsilon \tilde{M} = M(I + t v^T)$. Taking determinants and signs gives

$$\epsilon \doteq \det(I + t v^T) = 1 + v^T t = \hat{v}^T \hat{t}$$

as required. The same proof holds for ϵ' .

From (3) we have $H \hat{x}_i = \eta_i \hat{x}_i$. Multiplying this out and considering only the last component yields $\eta_i = k(v^T x_i + 1) = k \hat{v}^T \hat{x}_i$ as required. \square

Applying Lemma 8 to the case where one of the realizations is a strong realization leads to a necessary and sufficient condition for an epipolar configuration to be feasible.

Theorem 10. *Let $(P, P', \{x_i\})$ be any weak realization of an epipolar configuration $(Q, \{u_i\}, \{u'_i\})$ and let w_i and w'_i be defined as in (2). There exists a strong realization $(\bar{P}, \bar{P}', \bar{x}_i)$ of $(Q, \{u_i\}, \{u'_i\})$ if and only if $w_i w'_i$ has the same sign for all i .*

Proof. We begin by proving the if part of this theorem, and apply Lemma 8 to the case where $(\bar{P}, \bar{P}', \bar{x}_i)$ is a strong realization. In this case, $\bar{w}_i \doteq 1$ and so $w_i \eta_i \epsilon \doteq 1$. Similarly, $w'_i \eta'_i \epsilon' \doteq 1$. Therefore $w_i w'_i \eta_i^2 \epsilon \epsilon' \doteq 1$, from which it follows that $w_i w'_i \doteq \epsilon \epsilon'$ which is constant for all i .

Now, we turn to prove the converse. Let X^+ be the set of points x_i such that $w_i > 0$ and let X^- be the set of points such that $w_i < 0$. The sets X^+ and X^- are separated by the meridian planes of each of the cameras. Now, we seek a plane that separates X^- from X^+ and satisfies the additional condition that the perspective centres of the two camera lie on the same side of the plane if $w_i w'_i > 0$ for all i , or on opposite sides of the plane if $w_i w'_i < 0$ for all i . Such a plane can easily be found by slightly displacing the meridian plane of one of the cameras².

Let this separating plane be represented by a 4-vector \hat{v} . The condition that both perspective centres t and t' lie on the same or opposite sides of the plane may be written as $\hat{v}^T \hat{t} \doteq \kappa$ and $\hat{v}^T \hat{t}' \doteq \kappa w_i w'_i$ where κ is some non-zero value and $\text{sign}(w_i w'_i)$ is a constant for all i by hypothesis. The condition that the plane \hat{v} separates X^- from X^+ may be written as $\hat{v}^T \hat{x}_i \doteq \xi w_i$ for some constant ξ . Now, let H be the matrix

$$H = \begin{pmatrix} I & 0 \\ \kappa \xi v^T & \kappa \xi \end{pmatrix}.$$

Then according to Lemma 9, $\epsilon \doteq \hat{v}^T \hat{t} \doteq \kappa$, $\epsilon' \doteq \hat{v}^T \hat{t}' \doteq \kappa w_i w'_i$ and $\eta_i \doteq \kappa \xi \hat{v}^T \hat{x}_i \doteq \kappa \xi^2 w_i$. Now substituting into the equation $w_i \bar{w}_i \epsilon \eta_i \doteq 1$ from Lemma 8 yields $w_i \bar{w}_i \kappa^2 \xi^2 w_i \doteq 1$ from which it follows that $\bar{w}_i \doteq 1$ as required. Similarly, from the equation

²For this construction to work, it seems necessary to make the additional assumption that the point set $\{u_i\}$ is bounded in the image plane. This assumption will be true for any reasonable pinhole camera, which can not have an image of infinite extent.

$w'_i \bar{w}'_i \epsilon' \eta'_i \doteq 1$ we derive $w'_i \bar{w}'_i \kappa w_i w'_i \kappa \xi^2 w_i \doteq 1$, from which it follows that $\bar{w}_i \doteq 1$. This shows that $(\bar{P}, \bar{P}', \{\bar{x}_i\})$ is a strong realization as required. \square

Since the epipolar configuration derived from two images of a real scene must have a strong realization, this theorem gives a necessary and sufficient condition for a set of image correspondences to be realizable as a three dimensional scene. Theorem 10 is illustrated in Fig 2.

For planar object sets, Theorem 6 established the existence of a good projectivity between the object plane and the image plane. For non-planar objects seen in two views, strong realizations of the epipolar configuration take the rôle played by sets of image points in the two dimensional case.

Theorem 11. *Let $(Q, \{u_i\}, \{u'_i\})$ be an epipolar configuration and let $(P, P', \{x_i\})$ and $(\bar{P}, \bar{P}', \{\bar{x}_i\})$ be two separate strong realizations of the configuration. Then the projectivity mapping each point x_i to \bar{x}_i is good.*

Proof. With notation as in (3), $w_i \doteq \bar{w}_i \doteq 1$, and hence from Lemma 8, $\eta_i \epsilon \doteq 1$, which means that all η_i have the same sign. Therefore, by Theorem 5, H is a good projectivity. \square

The particular case where one of the two realizations is the "correct" realization is of interest. It is the analogue in three dimensions of Proposition 6.

Corollary 12. *If $\{x_i\}$ are points in R^3 , image coordinates $\{u_i\}$ and $\{u'_i\}$ are corresponding image points in two uncalibrated views, Q is the essential matrix derived from the image correspondences $u_i \leftrightarrow u'_i$ and $(P, P', \{\bar{x}_i\})$ is a strong realization of the triple $(Q, \{u_i\}, \{u'_i\})$, then there is a good projectivity taking each x_i to \bar{x}_i .*

From this corollary, we can deduce one of the main results of this paper.

Theorem 13. *Let $(P, P', \{x_i\})$ and $(\bar{P}, \bar{P}', \{\bar{x}_i\})$ be two different reconstructions of 3D scene geometry derived as strong realizations of possibly different epipolar configurations corresponding to possibly different pairs of images of a 3D point set. Then there is a good projectivity mapping each point x_i to \bar{x}_i .*

What this theorem is saying is that if a point set in R^3 is reconstructed as a strong realization from two separate pairs of views, then the two results are the same up to a good projectivity.

Proof. By corollary 12 there exist good projectivities mapping each of the sets of reconstructed points $\{x_i\}$ and $\{\bar{x}_i\}$ to the actual 3D locations of the points. The result follows by composing one of these projectivities with the inverse of the other. \square

7 Orientation

We now consider the question of image orientation. A mapping h from R^n to itself is called orientation-preserving at a point x if the Jacobian of h has positive determinant at x . Otherwise h is called orientation-reversing. Reflection of points of R^n with respect to a hyperplane (that is mirror imaging) is an example of an orientation-reversing mapping. A projectivity h from P^n to itself restricts to a mapping from $R^n - h^{-1}(L_\infty)$ to R^n , where L_∞ is the hyperplane (line, plane) at infinity. Consider the case $n = 3$ and let H be a 4×4 matrix representing the projectivity h . We wish to determine at which points x in $R - h^{-1}(L_\infty)$ the map h is orientation preserving. It may be verified (quite easily using Mathematica [Wolfram-88]) that if $H\hat{x} = w\hat{x}'$ and J is the Jacobian of h evaluated at x , then $\det(J) = \det(H)/w^4$. This gives the following result.

Proposition 14. *A projectivity h of P^3 represented by a matrix H is orientation preserving at any point in $R^3 - h^{-1}(L_\infty)$ if and only if $\det(H) > 0$.*

Of course, the concept of orientability may be extended to the whole of P^3 , and it may be shown that h is orientation-preserving on the whole of P^3 if and only if $\det(H) > 0$. The essential feature here is that as a topological manifold, P^3 is orientable. The situation is somewhat different for P^2 , which is not orientable as a topological space. In this case, with notation similar to that used above, it may be verified that $\det(J) = \det(H)/w^3$. Therefore, the following proposition is true.

Proposition 15. *A projectivity h of P^2 is orientation preserving at a point u in $R^2 - h^{-1}(L_\infty)$ if and only if $w \det(H) > 0$, where $Hu = w\hat{u}'$.*

This theorem allows us to strengthen the statement of Theorem 5 somewhat.

Corollary 16. *If h is a good projectivity of P^2 with respect to a set of points $\{u_i\}$ in R^2 , then h is either orientation-preserving or orientation-reversing at all points u_i . Suppose the matrix H corresponding to h is normalized to have positive determinant (by possible multiplication by -1) and let $Hu_i = w_i\hat{u}_i'$. Then h is orientation-preserving if and only if $w_i > 0$ for all i .*

An example where Corollary 16 applies is in the case where two images of a planar object are taken from the same side of the object plane. In this case, an orientation-preserving good projectivity will exist between the two images. Consequently, all the w_i defined with respect to a matrix H will be positive, provided that H is normalized to have positive determinant.

The situation in 3-dimensions is rather more involved and more interesting. Two sets of points $\{x_i\}$ and $\{\bar{x}_i\}$ that correspond via a good projectivity are said to be *oppositely oriented* if the projectivity is orientation-reversing. This definition extends also to two strong realizations $(P, P', \{x_i\})$ and $(\bar{P}, \bar{P}', \{\bar{x}_i\})$ of a common epipolar configuration $(Q, \{u_i\}, \{u_i'\})$,

since in view of Theorem 11 the point sets are related via a good projectivity. Whether or not oppositely oriented strong realizations exist depends on the imaging geometry. Common experience provides some clues here. In particular a stereo pair may be viewed by presenting one image to one eye and the other image to the other eye. If this is done correctly, then the brain perceives a 3-D reconstruction of the scene (a strong realization of the image pair). If, however, the two images are swapped and presented to the opposite eyes, then the perspective will be reversed – hills become valleys and vice versa. In effect, the brain is able to compute two oppositely oriented reconstructions of the image pair. It seems, therefore, that in certain circumstances, two oppositely oriented realizations of an image pair exist. It may be surprising to discover that this is not always the case, as is shown in the following theorem.

Theorem 17. *Let $(Q, \{u_i\}, \{u_i'\})$ be an epipolar configuration and let $(P, P', \{x_i\})$ be a strong realization of $(Q, \{u_i\}, \{u_i'\})$. There exists a different oppositely oriented strong realization $(\bar{P}, \bar{P}', \{\bar{x}_i\})$ if and only if there exists a plane in R^3 such that the perspective centres of both cameras P and P' lie on one side of the plane, and the points x_i lie on the other side.*

Before proving this theorem, we need a lemma.

Lemma 18. *Let $(P, P', \{x_i\})$ be a strong realization of an epipolar configuration $(Q, \{u_i\}, \{u_i'\})$. Then there exists a similarly oriented strong realization $(\bar{P}, \bar{P}', \{\bar{x}_i\})$ for which $\bar{P} = (I | 0)$.*

Proof. Suppose $P = (M | -Mt)$, with $\det(M) > 0$. Then multiplication by the matrix

$$H = \begin{pmatrix} M^{-1} & t \\ 0 & 1 \end{pmatrix}$$

transforms P to the required form. Furthermore, H^{-1} defines an orientation-preserving good projectivity on the points x_i . \square

Now, we may prove the theorem.

Proof. (Theorem 17) In light of Lemma 18 it may be assumed that P' and \bar{P}' are both of the form $(I | 0)$, because an oppositely oriented pair of realizations exist if and only if an oppositely oriented pair exist satisfying this additional condition.

Let us assume that such an oppositely oriented pair of strong realizations exists and H represents the orientation-reversing good projectivity relating them. We define ϵ , ϵ' and η_i as in (3). If necessary, H may be multiplied by a constant so that $\epsilon' = 1$. Since $w_i = w_i' = \bar{w}_i = \bar{w}_i' = 1$, it follows from Lemma 8 that $\eta_i = 1$ for all i and $\epsilon = 1$. From the equation $(I | 0)H = (I | 0)$ the form of H may be deduced:

$$H = \begin{pmatrix} I & 0 \\ k\nu^T & k \end{pmatrix}$$

for some 3-vector \mathbf{v} and, since H is orientation reversing, $k \doteq -1$.

Now, according to Lemma 9, $\eta_i \doteq k \hat{\mathbf{v}}^T \hat{\mathbf{x}}_i$, and since $\eta_i \doteq 1$ and $k \doteq -1$ it follows that $\hat{\mathbf{v}}^T \hat{\mathbf{x}}_i \doteq -1$. This condition may be interpreted as meaning that all the \mathbf{x}_i lie on one side of the plane defined by $\hat{\mathbf{v}}$.

On the other hand, by applying Lemma 9, we get $\hat{\mathbf{v}}^T \hat{\mathbf{t}} \doteq \epsilon \doteq 1$ and $\hat{\mathbf{v}}^T \hat{\mathbf{t}}' \doteq \epsilon' \doteq 1$. These equations mean that \mathbf{t} and \mathbf{t}' lie on the opposite side of the plane $\hat{\mathbf{v}}$ from all the points \mathbf{x}_i . This completes the *only if* part of the proof.

The converse may be proven by working backwards through this proof. Assuming the existence of a separating plane $\hat{\mathbf{v}}$ one constructs the orientation reversing matrix H as above and verifies that the resulting $(P, P', \{\hat{\mathbf{x}}_i\})$ is a strong realization. \square

Note that the existence of such a separating plane as described in Theorem 17 may be checked using any strong realization.

8 3D cheirality invariants

The cheirality invariant of a set of points may be computed from two views by constructing a strong realization of the epipolar configuration and then invoking Theorem 13. If in addition each pair of views is discovered to satisfy the condition of Theorem 17 then the orientation of the set of points with respect to a canonical basis gives a further invariant.

In general, finding a strong realization involves substantial computation. It is therefore convenient to be able to compute the cheirality invariant of a set of points from a weak realization. This may be done using the following theorem

Theorem 19. Suppose $(P, P', \{\mathbf{x}_i\})$ is a weak realization of an epipolar configuration $(Q, \{\mathbf{u}_i\}, \{\mathbf{u}'_i\})$ and let constants $\tilde{\eta}_i$ be defined for each \mathbf{x}_i as in the definition of the cheiral invariant. Suppose that $P\hat{\mathbf{x}}_i = w_i \hat{\mathbf{u}}_i$ and define $\eta_i = \tilde{\eta}_i w_i$, then $|\sum_{i=1}^N \text{sign}(\eta_i) 3^{i-1}|$ is the cheiral invariant of a strong realization of $(Q, \{\mathbf{u}_i\}, \{\mathbf{u}'_i\})$.

Details of the proof will not be given. It is simply a matter of considering the composition of two projectivities: from the strong realization to the weak realization and from the weak realization to the canonical frame.

9 Experimental results

In considering real images of 3-D configurations it is necessary to take into account the effects of noise. In particular, because of measurement inaccuracies, it will (virtually) never be the case that a point \mathbf{x}_i in a strong realization will map by chance exactly onto the plane at infinity under the mapping to the canonical basis. For this reason, in practical experiments I have preferred to define the cheiral invariant by interpreting the values η_i as bits of a binary integer: $\eta_i > 0$ corresponds to a 1 bit and $\eta_i < 0$ to a 0 bit. In some cases, a value of η_i will lie so close to 0 variations due to noise can swap its sign. For robust evaluation of a cheiral invariant value, it is necessary to select a noise model

and determine how errors in the input data affect the sign of each η_i . An investigation of noise propagation is underway with the purpose of assigning a computed error bound to each value η_i . The developed methods have not been implemented at present, so in the following discussion, noise effects are ignored.

In [Hartley-93] projective invariants of 3D point sets were discussed. As an experiment in that paper, a set of images of some model houses were acquired. Figures 1, 2 and 3 of [Hartley-93] show the three images. Corresponding vertices were selected by hand from among those detected automatically. The 13 vertices used are shown in [Hartley-93], Fig 4.

Six sets of six points were chosen as in the following table which shows the indices of the points as given in [Hartley-93], Fig 4.

$$\begin{aligned} S_1 &= \{1, 2, 3, 6, 9, 10\} \\ S_2 &= \{2, 4, 6, 8, 10, 12\} \\ S_3 &= \{1, 3, 5, 7, 9, 11\} \\ S_4 &= \{1, 2, 3, 6, 7, 8\} \\ S_5 &= \{1, 4, 7, 10, 13, 12\} \\ S_6 &= \{2, 5, 8, 11, 12, 13\} \end{aligned}$$

From image correspondences in two views ([Hartley-93], Figs 1 and 2) the essential matrix Q was found and a weak realization $(P, P', \{\mathbf{x}_i\})$ was computed. For each of the six sets of indices i shown above a complete projective invariant of the points $\{\mathbf{x}_i\}$ was computed by mapping the first five points onto a canonical basis. The coordinates of the mapped sixth point constitute a projective invariant of the set of six points.

This computation was repeated with a different pair of views ([Hartley-93], Figs 2 and 3). Theory predicts that the invariants should have the same value when computed from different views, and should distinguish between non-equivalent point sets.

Table (4) shows the comparison of the computed invariant values.

0.026	0.970	0.975	0.619	0.847	0.823
0.995	0.015	0.064	0.841	0.252	0.548
0.967	0.066	0.013	0.863	0.276	0.516
0.617	0.830	0.873	0.016	0.704	0.752
0.861	0.238	0.289	0.708	0.005	0.590
0.828	0.544	0.519	0.719	0.574	0.026

(4)

The (i, j) -th entry of the table shows the distance according to an appropriate metric between the invariant of set S_i as computed from the first image pair with that of set S_j as computed from the second image pair. The diagonal entries of the matrix (in bold) should be close to 0.0, which indicates that the invariants had the same value when computed from different pairs of views.

Although the projective invariants computed here are quite effective at discriminating between different point sets, indicated by the fact that most off-diagonal entries are not close to zero, entries (2, 3) and (3, 2) are small indicating that the point sets numbered 2 and 3 are close to being equivalent up to projectivity.

Next, the cheirality invariants for each of the point sets were computed from the weak realization using

the method described here. The computed values for each of the six point sets were as follows : $\chi(S_1) = 28$, $\chi(S_2) = 3$, $\chi(S_3) = 59$, $\chi(S_4) = 60$, $\chi(S_5) = 21$, $\chi(S_6) = 27$. As expected these invariant values were the same whether computed using the first pair of views or the second pair. Note that the cheirality invariant clearly distinguishes point sets 2 and 3. In fact, all six point sets are distinguished.

Reordering : Although there are no invariants of projectivity for 5 points in P^3 , the cheirality invariant is defined. In order to estimate its effectiveness for distinguishing different configurations the following experiment was carried out. Five points in P^3 were selected and the cheirality invariant computed for all permutations of the five points. The result was that 10 different invariant values were found (out of 16 possible), each one occurring 12 times. It may be seen that this will be true whichever 5 points are selected (though the invariant values will be different). In short, there is about one chance in 10 that two sets of five arbitrarily selected points will have the same cheirality.

When this experiment was carried out with 6 points arbitrarily chosen the results were seen to vary according to the particular configuration of the points. For various choices of points it was seen that the probability of getting a chance match for arbitrary permutations of the point set is about one chance in 20 or 30.

Conclusions : These results show that the cheirality invariant is quite effective at distinguishing between arbitrary sets of points. Given the relative ease with which the cheirality invariant may be computed, it may be extremely useful in grouping points. In addition, it may conveniently be used as an indexing function in an object recognition system. It has been demonstrated that the cheirality invariant gives supplementary information that is not available in projective invariants. As a theoretical tool, the cheirality invariants provide conditions under which image point matches may be realized by real point configurations.

References

- [Faugeras-92] Faugeras, O., "What can be seen in three dimensions with an uncalibrated stereo rig?", Proc. of ECCV-92, G. Sandini Ed., LNCS-Series Vol. 588, Springer-Verlag, 1992, pp. 563 - 578.
- [Higgins-81] H.C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," Nature, Vol. 293, 10 Sept. 1981.
- [Hartley-92] R. Hartley, "Invariants of 3D point sets", submitted for publication.
- [Hartley-Gupta-92] R. Hartley, R. Gupta and T. Chang, "Stereo from Uncalibrated Cameras", Proceedings Computer Vision and Pattern Recognition conference (CVPR-92), 1992.
- [Hartley-93] R. Hartley, "Invariants of Lines in Space", In these proceedings.
- [Wolfram-88] Wolfram, S., Mathematica, "A System for Doing Mathematics by Computer," Addison-Wesley, Redwood City, California, 1988.
- [Rothwell-92] Charles A. Rothwell, Andrew Zisserman, David A. Forsyth and Joseph L. Mundy, "Canonical Frames for Planar Object Recognition", Proc. of ECCV-92, G. Sandini Ed., LNCS-Series Vol. 588, Springer-Verlag, 1992, pp. 757-772.
- [Sparr-92] Gunnar Sparr, "Depth Computations from Polyhedral Images", Proc. of ECCV-92, G. Sandini Ed., LNCS-Series Vol. 588, Springer-Verlag, 1992, pp. 378 - 386.

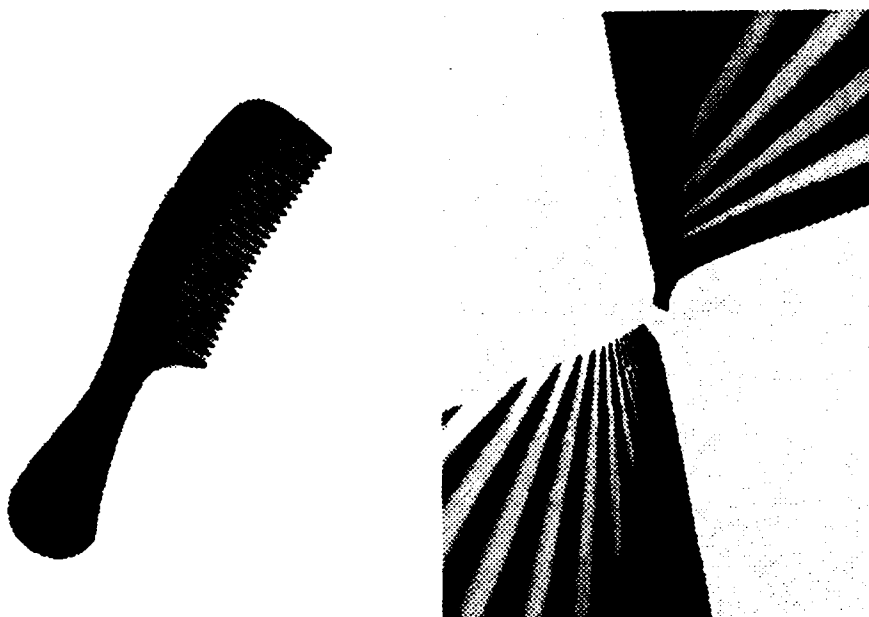


Figure 1. At the left a comb. At the right a naughty projection of the comb.

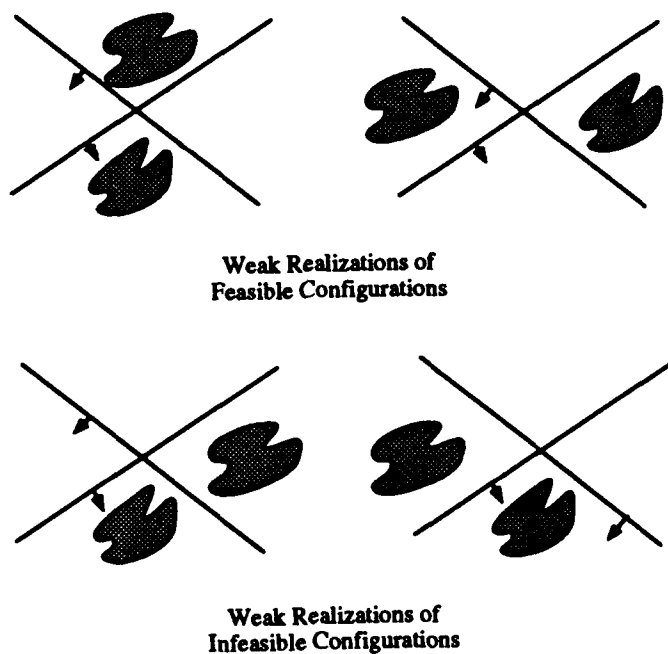


Figure 2. Each camera is shown symbolically as a line representing the meridian plane and an arrow indicating the direction of the front of the camera. Each diagram represents a weak realization of an epipolar configuration. The two top configurations of points and cameras satisfy the condition of Theorem 10 and may be converted to strong realizations. The two lower configurations do not, and hence can not be weak realizations of a real scene.

Efficient Recognition of Rotationally Symmetric Surface and Straight Homogeneous Generalized Cylinders

Jane Liu*

GE Corporate Research and Development
Schenectady, NY 12301

David Forsyth

Department of Computer Science
University of Iowa

Joe Mundy

GE Corporate Research and Development
Schenectady, NY 12301

Andrew Zisserman

Department of Engineering Science
Oxford University

Charlie Rothwell

Department of Engineering Science
Oxford University

Abstract

It is known that rotationally symmetric surfaces can be recognized from their outlines alone, using cross-ratios of bitangent intersections. This paper demonstrates a successful implementation of this technique, using a novel bitangent finder, that works on images of real scenes. The technique is shown to extend to the case of straight homogeneous generalised cylinders, and a dual construction for computing further invariants from outlines is displayed.

1. This paper is about recognizing SHGC's and rotationally symmetric objects, using outlines obtained from a single view by an uncalibrated camera, at an unknown viewing position.
2. This paper demonstrates a system that works, built using theory described in another paper (proc ECCV'92). It shows how this theory naturally extends to SHGC's, from rotationally symmetric objects. It then demonstrates constructions that yield further information about the surface, based on the dual of a surface.
3. One section briefly describes material already published, for background information only, and some of the paragraphs in the introduction appear in a paper submitted to the International Conference on Computer Vision; all other material is original.

1 Introduction

Outlines are a potentially important source of information about the objects in a scene because image edges appear at most outline points, and image edges can be computed reasonably reliably. This potential

has not been realised in the case of curved surfaces, because the outline of a curved surface is extremely hard to interpret. This paper demonstrates how a useful range of descriptions for rotationally symmetric surfaces and for straight homogeneous generalised cylinders can be determined from an outline in a single image, using simple geometric arguments. These descriptions are invariant to camera position, orientation and calibration.

1.1 Description for recognition

A number of recent papers have shown how projective or affine invariants can be used to index a model, and thereby avoid searching a model base (e.g. [10, 47, 35, 21]). To be used for indexing, a function must:

1. be computable from image outline information alone,
2. ideally should have different values for different objects and
3. be unaffected by object pose and intrinsic parameters of the camera.

Functions with these properties have the same value for any view of a given object, and so can be used to index into a model base without search; by abuse of terminology, we call such functions "indexing functions". Indexing functions have been demonstrated for polyhedra [37] (derived, as always from a single, unknown view), and for rotationally symmetric surfaces [11] (again, computed from an outline in a single, unknown view). A companion paper [12] proves the remarkable fact that a single outline yields all the projective geometry of an algebraic surface, and so demonstrates that all the projective invariants of an algebraic surface can, in principle, be computed for a single image. In this paper, we demonstrate successful indexing for a range of images of real scenes using

*Work at GE as supported in part by the DARPA under Contract No. MDA972-91-C-0053

the constructions of [11]. We show that these constructions also yield indexing functions for straight homogeneous generalised cylinders, and we demonstrate how further indexing functions may be obtained using the concept of the *dual surface*.

In a typical recognition system for planar objects, projective invariants¹ are computed for a range of geometric primitives in the image. If the values of these invariants match the values of the invariants for a known model, we have good evidence that the image features are within a camera transformation of the model features. Object models consist of sets of invariant values and are therefore relatively sparse, meaning that hypothesis verification is required to confirm a model match. However, no searching of the model base is required because the hypothesised object's identity is determined by the invariant descriptors measured. As a result, systems with relatively large model bases can be constructed². Systems of this sort have been demonstrated for plane objects in a number of papers [10, 36, 49, 21, 42, 48].

This paper concentrates on the more difficult problem of recognising curved surfaces from a single outline. Previous approaches include attempts to extend line labelling [13, 22], the development of constraint-based systems [4], and the study of how the topology of a surface's outline changes as it is viewed from different points, formalised into a structure known as an *aspect graph* (for example, [18, 19, 27, 33, 34]). Aspect graphs can be extremely complicated for even simple curved surfaces; some examples appear in [33, 34]. Recently, there have been attempts to represent the system of outlines of a curved surface as a linear combination of some small number of outlines (see, for example, [1, 2]). This approach is represented as providing an approximation sufficiently accurate for some purposes, although it cannot capture all the complexities that the aspect graph does.

Another area that has been extensively studied is the relationship between the differential geometry of the outline and of the surface, both for single images [19, 23] and for motion sequences [14, 6, 46]. It is generally accepted that the problem of recognising a surface from its outline alone is intractable if the surface is constrained only to be smooth, or piecewise smooth, as in this case significant changes can be made to the surface geometry without affecting the outline from a given viewpoint. As a result, an important part of the problem involves constructing as large a class of surfaces as possible that can either be directly recognised, or usefully constrained, from their outline alone. Dhome *et al.* showed that for a class of rotationally symmetric surface, object pose could be recovered for a known, calibrated camera, and incorporated this fact into a recognition scheme [7], which was later ex-

tended to include straight homogeneous generalised cylinders [8]. Relationships between sections of the outline of a straight homogeneous generalised cylinder have been widely studied, and are known to yield a variety of surface parameters in orthographic views [28, 44, 45].

In the case of plane objects, indexing functions are easy to compute, because viewing a plane curve from an arbitrary focal point induces an action of the projective group on the curve. Constructing indexing functions for three dimensional objects is challenging, because it is more difficult to ensure that these functions are *computable from outline information alone*, as changing viewing position no longer induces a group action on the outline.

1.2 The outline and its geometry

Throughout the paper, we assume an idealized pin-hole camera. These cameras possess a *focal point* and an *image plane*. For each point in space, there is a line through that point and the focal point; the point in space appears in the image as the intersection of this line with the image plane - figure 1 illustrates such a camera.

It is easy to see that if the focal point is fixed and the image plane is moved, the resulting distortion of the image is a collineation³. In what follows, it is assumed that neither the position of the image plane with respect to the focal point nor the size and aspect ratio of the pixels on the camera plane is known⁴, so that the image presented to the algorithm is within some arbitrary collineation of the "correct" image. In this abstract model, the image plane makes no contribution to the geometry, and its position in space is ignored. Notice that an orthographic view occurs when the pinhole is "at infinity".

The *outline* of a surface is a plane curve in the image, which itself is the projection of a space curve, known as a *contour generator*⁵. The contour generator is given by those points on the surface where the surface turns away from the image plane; formally, the ray through the focal point to the surface is tangent to the surface. As a result, at an outline point, if the relevant surface patch is visible, nearby pixels in the image will see vastly different points on the surface, and so outline points usually have sharp changes in image brightness associated with them. Figure 1 illustrates these concepts.

1.3 Indexing rotationally symmetric objects

It is shown in [11] that:

- **Lemma:** Except where the image outline cusps⁶, a plane tangent to the surface at a point on the

¹A clear introduction to applying invariant theory in computer vision appears in [24].

²Current systems using indexing functions have model-bases containing of the order of thirty objects.

³A collineation is a continuous, one-to-one map taking the projective plane to the projective plane which maps lines to lines; any collineation is a plane projective transformation.

⁴These quantities can be measured with varying degrees of difficulty; they do not appear to be particularly stable when cameras are moved, shaken or dropped, however.

⁵There are a number of widely used terms for both curves, and no standard terminology has yet emerged.

⁶We ignore cusps in the image outline in what follows.

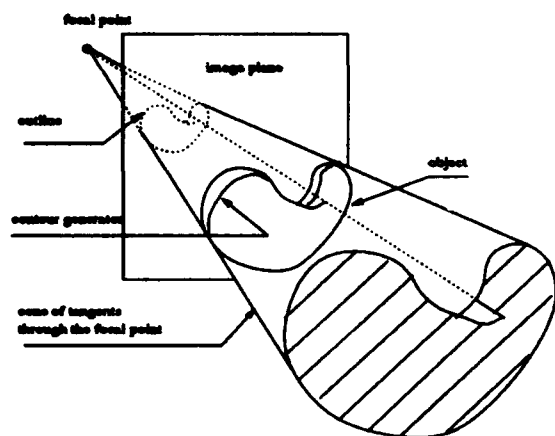


Figure 1: The outline and contour generator of a curved object, viewed from a perspective camera.

contour generator (by definition, such a plane passes through the focal point), projects to a line tangent to the surface outline, and conversely, a line tangent to the outline is the image of a plane tangent to the surface at the corresponding point on the contour generator.

- **Corollary 1:** A line tangent to the outline at two distinct points is the image of a plane through the focal point and tangent to the surface at two distinct points, both on the contour generator.
- **Corollary 2:** The intersection of two lines, bitangent to the outline is a point, which is the image of the intersection of the two bitangent planes represented by the lines.
- **Lemma:** For a rotationally symmetric surface, the envelope of the bitangent planes must be a right circular cone, or a cylinder (a cone with vertex at infinity).
- **Lemma:** The vertices of the cones bitangent to a rotationally symmetric surface must lie on the axis (by symmetry), and so are collinear. The vertices of the bitangent cones appear in the image as the intersections of a pair of lines bitangent to the outline; the image of the axis of the surface is a line passing through these bitangent intersections.
- **Indexing theorem:** Cross-ratios of corresponding image bitangent lines measure projective invariants of the surface. These projective invariants are cross-ratios of vertices of the bitangent cones which project to the bitangent lines. These invariants are determined from the outline alone. Furthermore, these image intersections can be used to construct the image of the axis of a rotationally symmetric surface from its outline.

Thus, cross-ratios of intersection points of corresponding bitangent lines yield indexing functions for rotationally symmetric surfaces. Note, in particular, that

these cross-ratios are *invariant to camera calibration*, and so can be used *with an unknown camera*. In section 2, we show how these cross-ratios can be computed reliably from image data, and demonstrate a simple recognition system using these cross-ratios; in section 3, we show that these cross-ratios can be used to index straight homogeneous generalised cylinders, and in section 4, we show a body of mathematical techniques that can be used to construct further indexing functions for rotationally symmetric surfaces and for straight homogeneous generalised cylinders.

2 A recognition system using cross-ratio's

A recognition system using cross-ratio's as indexing functions works as follows:

- cross-ratio's are constructed for corresponding pairs of bitangents in an image;
- these cross-ratio's are used as keys to a hash-table that contains the correspondence between surfaces and cross-ratio's to yield recognition hypotheses;
- the recognition hypotheses are tallied, verified and accepted or rejected.

In our existing system, we do not verify recognition hypotheses, as edge-based verification for curved surfaces is difficult without pose information, which is not available. The system's model-base contains three surfaces, and the system assumes that there is only one surface in each image to simplify the computation of corresponding bitangents.

The main step is computing cross-ratios from outlines. This process requires that:

1. all bitangents to the outline be found, and
2. corresponding bitangents identified and intersected.

2.1 Finding bitangent lines to a curve

A tangent line can be represented by Hough transformation as $l(\theta, \gamma)$, where θ is the orientation of the line and γ is the distance from the image center to the line, as shown in figure 2. Any line in the image can be mapped into a particular cell in the Hough transform table by its location and its orientation. If tangent lines derived from two different points fall into the same cell in the Hough transformation table, then those tangent lines are a bitangent line. The process of finding bitangents proceeds, therefore, by:

1. computing the tangents to the curve and Hough transforming these lines;
2. checking the Hough transformed system for cells containing more than one line, which are bitangents.

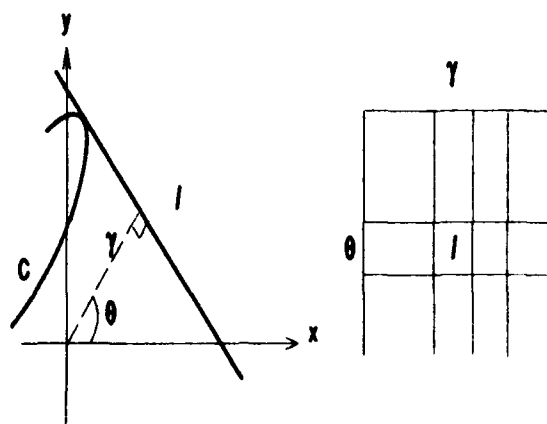


Figure 2: A Hough transformation table at the right is created with θ and γ indices. A tangent line to curve C is represented by θ and γ and stored in the cell (θ, γ) in the Hough transformation table.

2.1.1 Computing and Hough transforming tangents

Tangent lines are computed using an eigenvector line-fitting method [9]. As shown in the left half of figure 3, l is the best eigenvector fitting line based on the 7 points around P_i . In the experiments, we used an 11 point neighbourhood. The tangent line at P_i is the line passing through P_i and parallel to l , labelled t in the figure.

When the curve has high curvature, the θ and γ values of consecutive tangent lines can be quite different because of the sample spacing, with the result that two consecutive tangents can be marked in cells some way apart in the hough space. As a result, bitangent lines can be missed, because high-curvature segments of curves can lead to widely scattered points in the Hough space, which may not intersect properly (see the right half of figure 3 for an example). The solution to this problem is to interpolate between points in the Hough space, using either a linear or quadratic interpolate, depending on the variation in θ (for our experiments we used a quadratic interpolate if $\Delta\theta > 6^\circ$, and otherwise a linear interpolate). This strategy leads to continuous curves in the Hough space, and is successful in finding bitangents.

2.2 Determining corresponding bitangents

Once all bitangents have been found, it is necessary to determine which pairs of bitangents correspond (i.e. both come from the same cone of bitangents). This problem can be solved by exploiting the following remarkable symmetry property of rotationally symmetric surfaces:

Theorem: There is a non-trivial plane projectivity which maps the outline of a rotationally symmetric surface to itself. The contour generators corresponding to each half

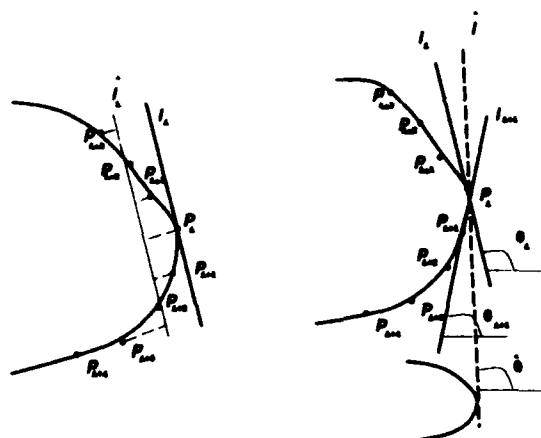


Figure 3: In the left half of the figure, an eigenvector fitting line l_i is constructed from seven points $P_{i-3}, P_{i-2}, P_{i-1}, P_i, P_{i+1}, P_{i+2}$, and P_{i+3} and the tangent line at point P_i is t_i , which passes through P_i and is parallel to l_i . In the right half, l is a bitangent line which is missed because l is not booked in the Hough transformation table while scanning P_i and P_{i+1} .

are, in general, space curves, and are related by a mirror symmetry in space.

In effect, this theorem is a stronger way of stating that the outline of a rotationally symmetric surface can be separated into two sides, which are related by plane projectivity. To see that the two sides of the contour in the image are projectively equivalent, for an arbitrary view, construct the plane containing the axis of the surface and the focal point. The surface then has a mirror symmetry in this plane, as does the cone of rays through the focal point and tangent to the surface. This cone yields the outline when it is intersected with the image plane.

If the image plane is perpendicular to the plane of symmetry, then the outline has a mirror symmetry; but the outline in any other image plane is within a projective map, say T of this outline (by construction, with the focal point as the centre of projection), and so we can construct a non-trivial projective mapping that takes the outline to itself as $P = T \circ M \circ T^{-1}$, where M is a mirror symmetry. Since by construction T is a projectivity, and M is a projectivity (it can be given as $\text{diag}[1, -1, 1]$), P is a projectivity.

This delivers a uniform method for determining points lying on the projection of the 3D symmetry axis. Any projectively covariant construction⁷ in the particular (symmetric) image plane which generates points on the image axis, can be used in any image. Examples include.

⁷By this, we mean that we would obtain the same result if we were to perform the construction in one frame, and then project the result to a new frame, or if we were to perform the construction in the new frame on a projection of the original curves: constructions with this property are based around incidence and counting properties. For example, a tangent line is a covariant construction.

1. Find corresponding pairs of distinguished points on each side of the outline, say a corresponding to a' , b corresponding to b' . Then the lines ab' , $a'b$ intersect on the symmetry axis. Appropriate distinguished points are covariants such as: points of contact of bitangents and inflections.
2. Determine the projective transformation that maps the contour to itself, and find the points that are fixed by this transformation. These points will form the projection of the axis. This construction might well be the best method (in a LMS sense), but has not yet been implemented.

In practice, we use approach 1 (above) with distinguished points derived from a bitangent's contact with the curve. Any pair of corresponding bitangents then generates two points on the axis image: one by the intersection of the bitangents (i.e. the lines ab and $a'b'$), the other by the *cross-construction* above (i.e. the lines ab' , $a'b$). This is a simple and successful construction. Note that the order of the points of tangency on each bitangent can be given with reference to their intersection point and so is uniquely defined.

Now, select any two bitangent lines in the image. We give a vote to line c , from both their intersection and cross-construction. The total number of votes for the correct image of the central axis, n , is the number of distinguished bitangent cones constructed by the shape of the object. The total number of votes for each incorrect image of the central axis clearly must be 1 or small if the surface is not degenerate, and so the line with maximum number of votes is the image of the real central axis. This voting system is refined further by noting that, for real views, it is extremely hard to arrange the camera such that corresponding bitangent lines are more than a few degrees away from parallel. Currently, pairs of bitangents where these lines are more than 4° off parallel do not contribute to the vote.

2.3 Results

In a total of 15 test images, the correct object was identified in each case. Recognition proceeded by computing all possible cross-ratios of bitangent intersections from an image, rounding these values to a single digit, and using them as a key to a hash-table, which was preloaded with the names of the surfaces, using cross-ratios computed from one image of each surface. Tables 1-3 show the details of the returns from the hash-table for a range of different images of different objects, and table 4 shows the data collated. In particular, for the stand and the doorknob, a number of cross-ratios could be computed from each image, and the final identification was made by voting for the object with the greatest number of returns. Note that the technique described is showing a degree of robustness, as surfaces are correctly identified despite the differing number of cross-ratios computed for each image as a result of noise-related difficulties in obtaining all bitangents.

view	lamp	ambiguous, including lamp	other	miss
1	1	0	0	0
2	1	0	0	0
3	1	0	0	0
4	1	0	0	0
5	1	0	0	0

Table 2: Hash-table returns for five views of a lamp, using a hash table preloaded using a sixth view of that surface, and views of two other surfaces. Note that the surface is clearly identified in each case by choosing the return with the most votes. The columns show the label returned from the hash-table; alternatives are the correct label, a number of labels including the correct label, a collection that does not include the correct label, and nothing at all.

view	stand	ambiguous, including stand	other	miss
1	4	0	0	0
2	4	0	0	0
3	4	0	0	0
4	3	0	0	0
5	1	0	0	0
6	2	0	0	0

Table 3: Hash-table returns for six views of a stand, using a hash table preloaded using a seventh view of that surface, and views of two other surfaces. Note that the surface is clearly identified in each case by choosing the return with the most votes. The columns show the label returned from the hash-table; alternatives are the correct label, a number of labels including the correct label, a collection that does not include the correct label, and nothing at all.

view	"doorknob"	ambiguous, including "doorknob"	other	miss
1	3	0	1	0
2	4	0	1	0
3	4	0	1	0
4	4	0	1	0

Table 1: Hash-table returns for four views of a doorknob, using a hash table preloaded using a fifth view of that surface, and views of two other surfaces. Note that the surface is clearly identified in each case by choosing the return with the most votes. The columns show the label returned from the hash-table; alternatives are the correct label, a number of labels including the correct label, a collection that does not include the correct label, and nothing at all.

surface	number of views	number correctly identified	failures
doorknob	4	4	0
lamp	5	5	0
stand	6	6	0

Table 4: Composite results of indexing surfaces from a range of views, showing the surfaces identified by a return from a hash-table, indexed by invariants computed from image information. Note that in each case, the vast majority of returns from the hash table either uniquely identify the correct surface, or contain the correct surface as an option in an ambiguous return. To identify surfaces, all returns are taken as votes for the surfaces returned, and the surface receiving the maximum number of votes is accepted. In no views, of a total of 15, was the final identification incorrect.

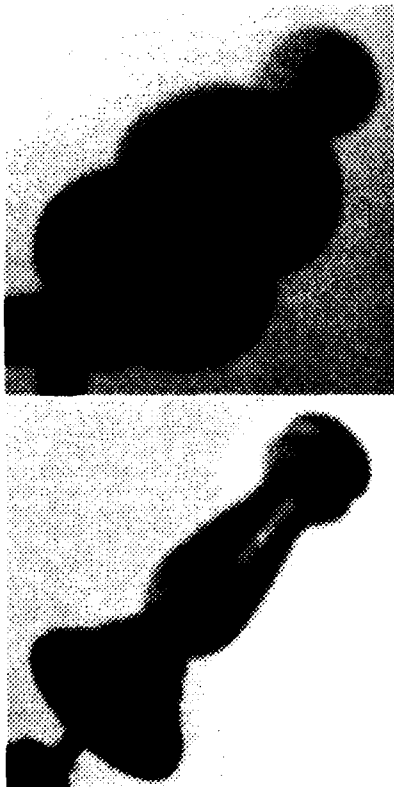


Figure 4: Typical images of real rotationally symmetric objects, used to obtain the recognition results; the top figure shows the knob, the lower figure shows the stand.

3 Indexing straight homogeneous generalised cylinders

A straight homogeneous generalised cylinder (SHGC) can be defined as a surface that, in some Euclidean frame, can be parametrised as:

$$(f_1(t)g_1(s), f_1(t)g_2(s), f_2(t))$$

Thus, in the appropriate frame, the sections of this surface corresponding to planes $z = \text{constant}$ are uniformly scaled copies of the plane curve $(g_1(s), g_2(s))$. As a result, in this frame the z -coordinate axis forms an "axis" for the surface, which has a similar role to the axis of a rotationally symmetric surface.

Now consider the family of planes through this axis; an arbitrary plane from this family is given by $ax + by = 0$, for some a, b . In coordinates in this plane, the intersection between the surface and the plane can be given by:

$$(\lambda f_1(t), f_2(t))$$

where λ is a function of s (figure 6). In particular, only λ changes as we move from plane to plane in the family. We have:

Lemma: The envelope of the family of planes tangent to the surface along a curve of fixed t (a "parallel"), is a cone or a cylinder.

The lemma is proven by noting that every tangent plane in this family intersects the z -axis in the same point; this, in turn is proven by showing that the y -intercept of a line tangent to a curve of the form $(\lambda f_1(t), f_2(t))$ is the same for any $\lambda \neq 0$. Note that the cones or cylinders are also SHGC's, with the z -axis

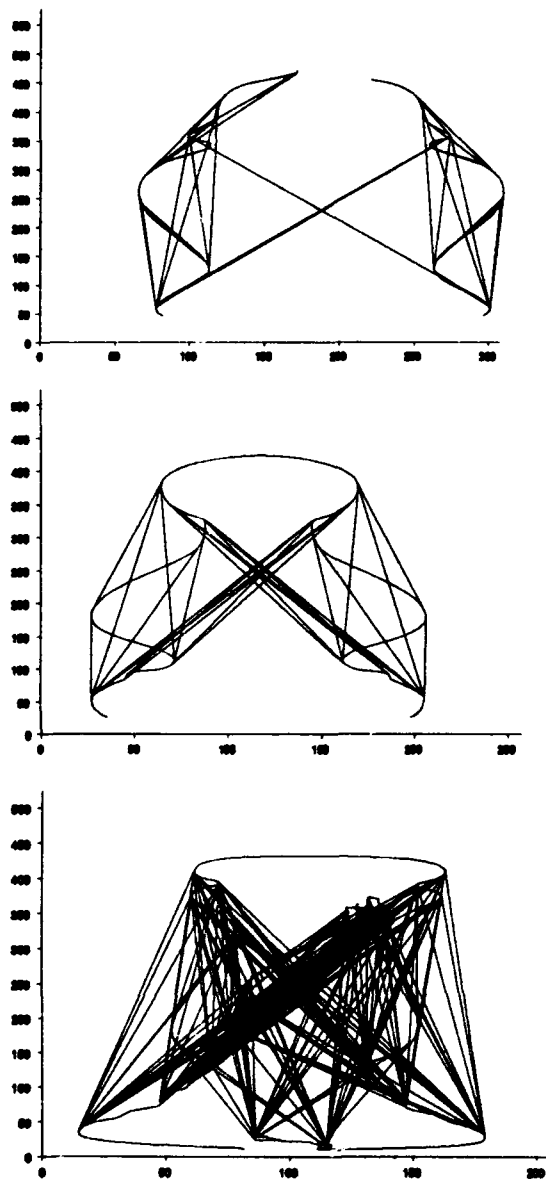


Figure 5: This figure shows all the constructed bitangent lines and the outlines of the images of three samples: a lamp (top), a knob and a candle stand (bottom).

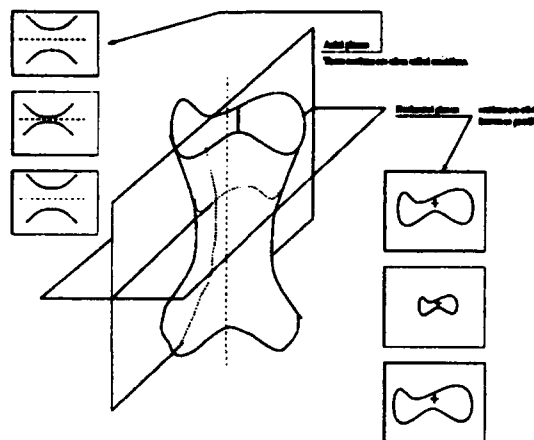


Figure 6: Plane sections of a straight homogeneous generalised cylinder, illustrating meridians and parallels.

as their "axis" and with the same cross-section as the surface.

From this lemma, we have immediately:

Lemma: Families of planes bitangent to SHGC's form cones, with their vertices on the axis of the surface.

Thus, we can construct indexing functions for SHGC's in exactly the same way as we constructed indexing functions for rotationally symmetric surfaces.

4 Using the dual to construct further indexing functions

The previous constructions have been shown to yield indexing functions for rotationally symmetric surfaces, which we have shown have genuine value for identifying the surface. Simple constructions like bitangent cones appear to yield no further invariants; for that, we must pass to the dual of the surface.

There is a natural duality between points in space and planes in space; a point is given by four homogeneous coordinates, and so is a plane. This duality can be extended to the case of surfaces, where the dual of a surface is defined to be the object given by the collection of points dual to the surface's tangent planes. For example:

- The dual of a plane is a point.
- The dual of a cone is a plane curve; to see this, note that the planes tangent to a cone all pass through its vertex, and hence all satisfy a single linear equation. Thus, all the points on the dual must satisfy a single linear equation, and so the dual must be a plane curve.
- The dual of a quadric surface given by the equation $\mathbf{x}^T \mathbf{Q} \mathbf{x} = 0$ is a quadric surface given by the equation $\mathbf{x}^T \mathbf{Q}^{-1} \mathbf{x} = 0$.

The dual has the following important property:

Theorem: The outline of a surface in a perspective view is equivalent to a plane section of the dual of that surface, where the sectioning plane is the plane dual to the focal point.

A version of this result has been well-known amongst geometers for a long time [5], but it does not appear to have diffused into the vision community to date. The result follows from the observation that the outline is, essentially, formed by the family of planes tangent to the surface and passing through the focal point. Thus, it is given by a family of planes tangent to the surface, and satisfying a single, linear relation. In turn, this means that the family is a plane section of the dual surface. We list below how taking duals affects a range of geometric concepts:

Define a projective rotationally symmetric (PRS) surface to be a surface that is within a 3D projectivity of a rotationally symmetric surface. Then the following is easily proven:

Theorem: The dual of a PRS surface is a PRS surface. In particular, if M is a meridian of the original PRS surface S taken in the frame in which it is rotationally symmetric, then the dual of S is within a 3D projectivity of a rotationally symmetric surface whose meridian is M .

4.1 Example: proving existing results

Here we rederive the results of section 1.3 using the concept of a dual surface. Notice that because we are dealing with a rotationally symmetric surface, the bitangents form cones; the duals of these cones are then plane curves in the dual space, where the plane the curve lies on is dual to the vertex of the cone (from above). Thus, there is a system of distinguished planes in the dual space, where the dual surface has self-intersections (dual to bitangents). All these planes have the further property that they are drawn from a single pencil of planes (the points to which they are dual are collinear). Since a pencil of planes is a one-parameter family of planes, parametrized by a line, these planes have a meaningful cross-ratio.

In a plane section of the dual, any self-intersections that cross the plane will be obvious as self-intersections of the section (figure 8). Note that for some sectioning planes, the singularities of the dual do not appear in the section, and this corresponds to those awkward viewing positions where the outline of a rotationally symmetric surface does not have bitangents - for example, a view down the axis. If we construct the lines connecting corresponding singularities, we obtain lines drawn from a pencil; but the cross-ratio of these lines is equivalent to the cross-ratio of the planes, and so is a projective invariant of the surface, that is invariant to choice of sectioning plane, as long as it can be observed.

4.2 A new invariant

Most of this discussion is based around the following useful lemma, which is dual to that giving a pro-

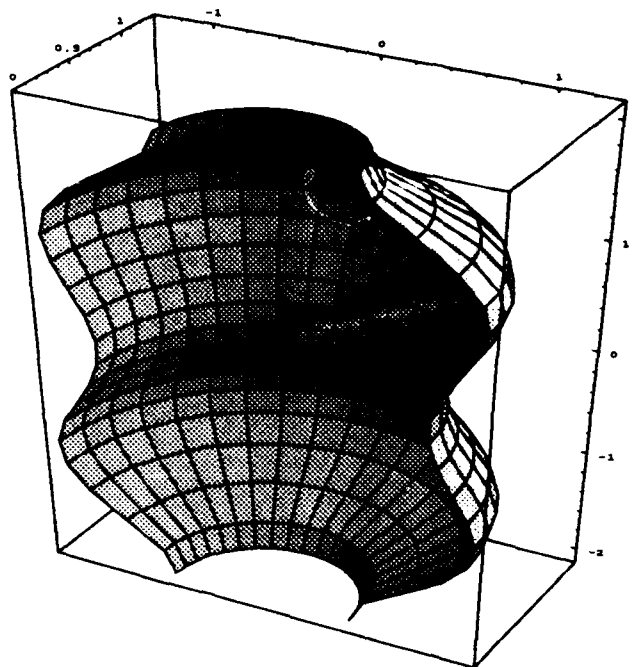


Figure 7: This figure shows a cut-away version of a rotationally symmetric surface; note the inflections and bitangents of the meridian.

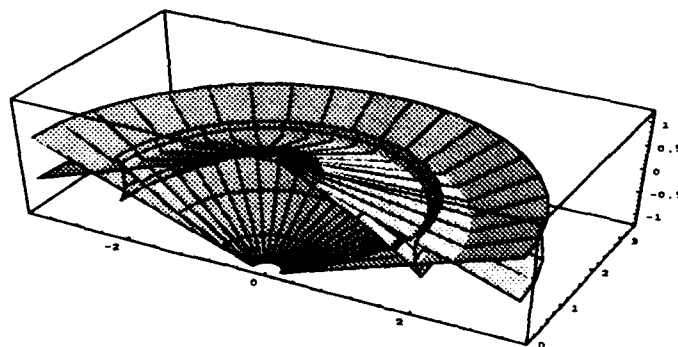


Figure 8: This figure shows a cut-away version of the dual of the surface in figure 7. Notice the cuspidal edges, corresponding to inflections of the meridian, and the self-intersections, corresponding to bitangent cones; all these singularities lie on parallels.

Original Space	Dual Space
incidence	tangency
tangency	incidence
plane	point
point	plane
line	line
cone of tangent planes through the focal point	plane cross section of the dual
bitangent plane	self-intersection
bitangent cone	plane curve of self-intersections
parabolic line	cuspidal edge
generic surface	surface with cuspidal edges and self intersections
order of contact with line	same order of contact with line
asymptotic curves	asymptotic curves
flecnodal curves	flecnodal curves
sign of K_G	sign of K_G

jectivity from the outline of a rotationally symmetric surface to itself:

Lemma: There is a non-trivial, plane projectivity that takes any arbitrary plane section of a PRS to itself.

To prove this, we work in a frame in which the surface is, in fact, rotationally symmetric. In this frame, construct a plane through the axis of the surface at right angles to the sectioning plane. The constructed plane yields a plane in space about which the surface and the sectioning plane are symmetric, and so yields a line in the sectioning plane about which the plane section has a symmetry. In turn, this symmetry (which is a projectivity of space, say, I) yields a map that takes the section to itself in the same way that the symmetry of an outline from a special viewing position yields a map that takes the outline to itself (section 2.2), that is, a map of the form $P^I - 1)IP$, where P and I are projectivities of space.

Given a surface is a PRS, its dual is a PRS too; now consider dual surface, which is PRS, and let us work in the frame in which this dual surface is rotationally symmetric, with the z -axis as its axis. For a general surface, the dual will have self-intersections which are circles lying on planes of constant z , and so we can construct a quadric which contains any three of these circles. In our frame, the quadric will have the equation:

$$x^2 + y^2 - (az^2 + bz + c) = 0$$

It is easy to see that this quadric exists, is unique for three distinct circles and can be constructed; furthermore, any intersections between the quadric and the dual surface will be circles, again lying on planes of constant z . Since the quadric is uniquely defined by incidence relations alone, the construction is projectively covariant, and so any cross-ratio's incorporating this quadric's intersection planes will be projective invariants.

To show that these invariants can be measured from a single, unknown image, we need to show that they can be measured from a single plane section of the dual. For this, work in a frame in which the

plane section's symmetry is expressed as $diag[-1, 1, 1]$; then any intersection between a rotationally symmetric quadric and the surface, that passes through three singularities, must appear in the section in the form $x^2 - ay^2 - by - c = 0$. Three incidence conditions determine this curve exactly, and since the result is unique, it must be a section of the unique quadric passing through the corresponding circles. Thus, the intersection points between this curve and the plane section correspond to intersection points between the dual and the quadric, and we are done.

Returning to non-dual space, the dual of the quadric intersecting the dual surface is again a PRS quadric, but here tangent to the original surface at the two circles of inflections, and tangent to the bitangent cone (again a circle of contact). The projection of this quadric in the image is the unique conic tangent at corresponding inflections and bitangent lines on both "sides" of the outline. New invariants can then be generated from this conic. For example, corresponding bitangents between this conic and the outline intersect on the axis in the same manner as bitangents of the outline.

These invariants can be constructed from image data, by taking the dual of the outline, which will have the features described, and performing the constructions described on that dual. It is not yet known whether more efficient algorithms exist, nor is it known how many independent invariants can be obtained in this way.

The lemma certainly allows many invariants to be constructed, either by constructing higher degree interpolants and using them in the same way the quadric was used, or by noting that, for any plane section of the dual, if the section is in the frame in which its symmetry is of the form $diag[-1, 1, 1]$, then points on the outline with the same y value correspond to the same parallel. If the parallel can be identified from plane section to plane section, for example, by the presence of a surface marking, a change in colour, or an incidence property (similar to those above), it can be used to generate cross-ratio's. Thus, a rather full invariant description of a rotationally symmetric surface is possible from a single outline.

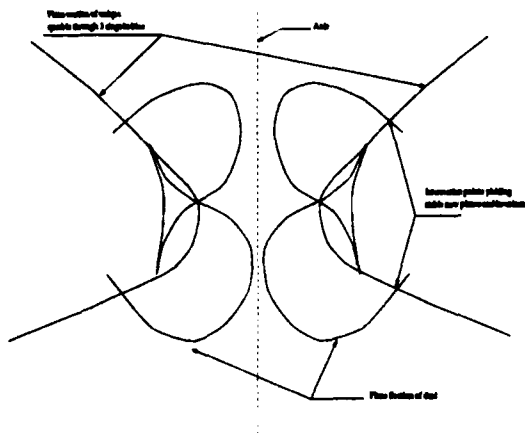


Figure 9: A plane section of a dual surface, showing three singularities on each side, with a quadric of the form $x^2 - ay^2 - by - c = 0$ superimposed. This is the unique quadric of this form that passes through the singularities; it generates two further intersection planes, which are shown, to yield a cross-ratio.

5 Discussion

We have demonstrated that rotationally symmetric surfaces can be successfully indexed using bitangents computed automatically from image edges by a bitangent finding algorithm which we have described. We have shown that this approach can be extended to recognise straight homogeneous generalised cylinders, and we have demonstrated techniques for constructing further indexing functions for rotationally symmetric surfaces.

References

- [1] Basri, R. and Ullman, S. "The alignment of objects with smooth surfaces," *Proc ICCV-2*, Tarpou Springs, 1988.
- [2] Basri, R. and Ullman, S. "Recognition by linear combination of models," *IEEE PAMI*.
- [3] Basset, A.B., *A treatise on the geometry of surfaces*, George Bell and Sons, London, 1910.
- [4] Brooks, R. A., "Model-Based Three-Dimensional Interpretations of Two Dimensional Images," *IEEE PAMI*, 5, 2, p. 140, 1983.
- [5] Bruce, J.W., "Lines, surfaces and duality", *Preprint, Dept. of Pure Mathematics, University of Liverpool*, 1992.
- [6] Blake, A. and Cipolla, R., "Robust estimation of surface curvature from deformation of apparent contours", In O. Faugeras, editor, *Proc. 1st European Conference on Computer Vision*, pages 465-474, Springer-Verlag, 1990.
- [7] Dhome, M., LaPrete, J.T., Rives, G., and Richetin, M. "Spatial localisation of modelled objects in monocular perspective vision," *Proc. First European Conference on Computer Vision*, O.D. Faugeras (ed.), Springer LNCS-x, 1990.
- [8] Dhome, M., Glachet, R. and LaPrete, J.T., "Recovering the scaling function of an SHGC from a single perspective view," *Proc. CVPR-92*, 1992.
- [9] Duda, R. O. and Hart, P. E., "Pattern Recognition Classification and Scene Analysis", a Wiley-interscience publication 1973.
- [10] D.A. Forsyth, J.L. Mundy, A.P. Zisserman, A. Heller, C. Coelho and C.A. Rothwell, "Invariant Descriptors for 3D Recognition and Pose," *IEEE Trans. Patt. Anal. and Mach. Intelligence*, 13, 10, 1991.
- [11] Forsyth, D.A., Mundy, J.L., Zisserman, A. and Rothwell, C.A., "Recognising rotationally symmetric surfaces from their outlines," *Proc. Second European Conference on Computer Vision*, G. Sandini (ed.), Springer LNCS-x, 1992.
- [12] Forsyth, D.A. "Recognizing algebraic surfaces from their outlines," submitted to *International Conference on Computer Vision*, 1992.
- [13] H. Freeman and R. Shapira, "Computer Recognition of Bodies Bounded by Quadric Surfaces from a set of Imperfect Projections," *IEEE Trans. Computers*, C27, 9, 819-854, 1978.
- [14] Giblin, P. and Weiss, R. *ICCV-1*, 1986.
- [15] Griffiths, P. and Harris, J. *Methods of Algebraic Geometry*, John Wiley and Sons, 1986.
- [16] Hartshorne, R. *Algebraic Geometry*, Springer Verlag Graduate Texts in Mathematics, 1977.
- [17] Kapur, D. and Lakshman, Y.N.
- [18] Koenderink, J.J. *Solid Shape*, MIT Press, 1990.
- [19] Koenderink, J.J. "What does the occluding contour tell us about Solid Shape," *Perception*, 13, 1984
- [20] Koenderink, J.J. and Van Doorn, A., "The Internal Representation of Solid Shape with respect to Vision," *Biological Cybernetics*, 32, 1979.
- [21] Lamdan, Y., Schwartz, J.T. and Wolfson, H.J. "Object Recognition by Affine Invariant Matching," *Proceedings CVPR*, p.335-344, 1988.
- [22] Malik, J., "Interpreting line drawings of curved objects," *IJCIV*, 1, 1987.
- [23] Marr, D., *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W.H. Freeman and co., San Francisco, 1982.
- [24] J.L. Mundy and A.P. Zisserman, "Introduction," in J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.
- [25] J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.

- [26] Ohmi, J. "Space curves as ideal-theoretic complete intersections," in Seidenberg, A. (ed), *Studies in algebraic geometry*, MAA Studies in Mathematics, 1980.
- [27] Plantinga, H. and Dyer, C. "Visibility, Occlusion and the Aspect Graph," CS TR 736, U. Wisconsin, 1987.
- [28] Ponce, J. "Invariant properties of straight homogeneous generalised cylinders," *IEEE Trans. Patt. Anal. Mach. Intelligence*, **11**, 9, 951-965, 1989.
- [29] Ponce, J. and Kriegman, D.J. "On Recognising and Positioning Curved 3 Dimensional Objects from Image Contours," *Proc. DARPA IU Workshop*, 1989.
- [30] Ponce, J. and Kriegman, D.J. "Computing exact aspect graphs of curved objects: parametric patches," *Proc. AAAI Conf.*, Boston, July, 1990.
- [31] Ponce, J. and Kriegman, D.J. "New progress in prediction and interpretation of line-drawings of curved 3D objects," *Proc 5th IEEE Int. Symp. Intelligent Control*, 1990.
- [32] Ponce, J., Hoogs, A. and Kriegman, D.J. "On using CAD models to compute the pose of curved 3D objects," *Proc IEEE workshop on Directions in Automated CAD-based Vision*, 1991.
- [33] Ponce, J. and Kriegman, D.J., "Toward 3D curved object recognition from image contours," in J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.
- [34] Rieger, J. "Global Bifurcation Sets and Stable Projections of Non-Singular Algebraic Surfaces," *Int. J. Computer Vision*, **7** 3, 1992.
- [35] Rothwell, C.A., Zisserman, A.P., Forsyth, D.A. and Mundy, J.L., "Using Projective Invariants for constant time library indexing in model based vision," *Proc. British Machine Vision Conference*, 1991.
- [36] Rothwell, C.A., Zisserman, A.P. Forsyth, D.A. and Mundy, J.L., "Fast recognition using algebraic invariants," in J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.
- [37] Rothwell, C.A., Forsyth, D.A., Zisserman, A. and Mundy, J.L. "Extracting Projective Information from Single Views of 3D Point Sets," *TR OUEL 1927/92*, Department of Engineering Science, Oxford University, Oxford, 1992.
- [38] Salmon, G. *Modern Higher Algebra*, Chelsea, New York.
- [39] Sederberg, T.W. "Techniques for cubic algebraic surfaces - part 1" *IEEE Computer Graphics and Applications*, July, 1990.
- [40] Sederberg, T.W. "Techniques for cubic algebraic surfaces - part 2" *IEEE Computer Graphics and Applications*, September, 1990.
- [41] Sugihara, K. *Machine Interpretation of Line Drawings*, MIT Press, Cambridge, 1986.
- [42] Taubin, G. and Cooper, D.B., "Object recognition based on moment (or algebraic) invariants," in J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.
- [43] Terzopolous, D., Witkin, A. and Kass, M. "Constraints on Deformable Models: Recovering 3D Shape and Nonrigid Motion," *Artificial Intelligence*, **36**, 91-123, 1988.
- [44] Ulupinar, F. and Nevatia, R. "Shape from Contour using SHGC's," *Proc. ICCV*, Osaka, 1990.
- [45] Ulupinar, F. and Nevatia, R. "Recovering shape from contour for constant cross-section generalised cylinders," *Proc. CVPR*, Maui, 1991.
- [46] Vaillant, R., "Using occluding contours for 3D object modelling", In O. Faugeras, editor, *Proc. 1st European Conference on Computer Vision*, pages 454-464, Springer-Verlag, 1990.
- [47] Wayner, P.C. "Efficiently Using Invariant Theory for Model-based Matching," *Proceedings CVPR*, p.473-478, 1991.
- [48] Weiss, I. "Projective Invariants of Shapes," *Proceeding DARPA Image Understanding Workshop*, p.1125-1134, April 1988.
- [49] Zisserman, A.P., Forsyth, D.A., Mundy, J.L. and Rothwell, C.A., "Recognizing general curved objects efficiently," in J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.

Projective Invariant and Structure from Two Perspective/Orthographic Views: Motion and Recognition

Amnon Shashua
Artificial Intelligence Laboratory
and Department of B&CS
M.I.T.
545 Technology Square
Cambridge, MA 02139

Abstract

We address the problem of reconstructing 3D space in a projective framework from two views, and the problem of artificially generating novel views of the scene from two given views. We show that with the correspondences coming from four non-coplanar points in the scene and the corresponding epipoles, one can define and reconstruct (using simple linear methods) a projective invariant that can be used later to reconstruct the projective or affine structure of the scene, or directly to generate novel views of the scene. The derivation has the advantage that the viewing transformation matrix need not be recovered in the course of computations (i.e., we compute structure without motion).

1 Introduction

This paper presents a study on the geometric relation between objects and their views (perspective and orthographic) geared towards developing tools with applications to 3D reconstruction and visual recognition. For this purpose we define a new projective invariant that can be computed from image measurements across two views (four corresponding points and the epipoles) using simple linear methods. The invariant is then used for reconstructing the 3D scene in projective or affine space, and for generating novel views of the scene/object directly — without going through projective coordinates and camera transformation.

We adopt the projective framework for representing 3D space as was also done recently by [6, 19, 11]. In a projective framework the scene is represented with respect to a frame of reference of five points whose location in space are unknown and can assume arbitrary general configurations in 3D projective space [29]. This allows us to work in a framework that does not make a distinction between orthographic and perspective views and does not require internal camera calibration, i.e., the internal camera parameters are folded into the camera transformations.

Related to 3D reconstruction is the application to visual recognition. The alignment approach to recognition [9, 18, 27, 13, 28] is based on the notion that the geometric relation between objects and their images can

be used to create an equivalence class of images of an object of interest. This approach can be realized by storing a few number of "model" views (two, for example) and with the help of corresponding points between the model views and any novel input view, the object is "re-projected" onto the novel viewing position. Recognition is achieved if the re-projected image is successfully matched against the input image. We refer to the problem of predicting a novel view from a set of model views using a limited number of corresponding points, as the problem of *re-projection*.

The problem of re-projection can in principal be dealt with via 3D reconstruction of shape and camera motion. For purposes of stability, however, it is worthwhile exploring more direct tools for achieving re-projection. Most of the current tools available for this purpose assume orthographic projection [28, 14, 22]. The method of epipolar line intersection is a possibility for achieving re-projection under perspective [3, 23] but, however, is singular for certain viewing transformations. For example, numerical instabilities arise when the centers of projection of the three cameras are nearly collinear, or equivalently, when the object rotates around nearly the same axis for all views. The re-projection method introduced in this paper is not based on an epipolar intersection, but rather is based directly on the relative structure of the object, and does not suffer from any singularities, a finding that implies greater stability in the presence of noise.

We derive a geometric invariant defined by a single cross ratio along a ray cutting through the frame of reference. The invariant can be used later to recover homogeneous coordinates if desired, or used directly to achieve re-projection onto a third view. The derivation has the advantage that the viewing transformation need not be recovered in the course of the computations — only the projections due to two faces of the tetrahedron of reference. The geometric construction we use requires the projections of four scene reference points onto two views, and as the fifth reference point we use the camera's center of projection via the epipoles. The epipoles are used both as a fifth corresponding pair and a means for determining correspondences due to projections of various faces of the tetrahedron of reference.

Part of this work originally appeared in [24] describing the geometric invariant and its application to re-

projection, and was derived independently of [6, 19, 11]. The later stage of reconstructing homogeneous coordinates given the recovered invariant is inspired by the work of [6].

2 Projective Framework and Related Work

In a projective framework the location of an object point is measured relative to a frame of reference of five points (a tetrahedron and a unit point) whose positions in space are unknown and which are allowed to map onto any general configuration of five points in 3D projective space. It is not difficult to show [25] that the space of images we can get out of this framework are no more than perspective and orthographic images of the scene, and images of images of the scene, produced by a pin-hole camera in which the camera's coordinate frame is allowed to undergo arbitrary affine transformations in space.

The projective framework enlarges the equivalence class of images of an object compared to the metric framework, but in return does not require internal camera calibration and does not make a distinction between orthographic and perspective projections. The internal camera parameters (focal length, principal point and image coordinates scale factors) are folded into the affine transformation of the camera coordinate frame ([20], for example) and, therefore, can assume arbitrary values (which can also change from one view to another). Orthographic images are included in this framework because any of the reference points (including the COP) can be anywhere in 3D projective space. These features of the projective framework imply greater stability in the presence of noise compared to the metric framework (see [1, 5, 26, 4, 23] for discussions on the performance of metric structure-from-motion in the presence of noise).

Projective space can be represented by homogeneous or non-homogeneous coordinates. In a non-homogeneous representation a point P is represented by three cross ratios along three axes of the tetrahedron of reference (see Figure 1). A homogeneous representation is a tetrad (x, y, z, t) of coordinates which is typically realized by assigning the standard coordinates $(0, 0, 0, 1), (1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (1, 1, 1, 1)$ to the vertices of the tetrahedron O, U, V, W and the unit point T , respectively (see Figure 2). For example, the points with $t = 0$ are on the plane UVW , and the projection of P via O is the point with coordinates $(x, y, z, 0)$ (i.e., orthographic projection in coordinate space). In general, any ordered set of four numbers, not all zero, determine uniquely a point in space.

A geometric reconstruction of non-homogeneous coordinates was recently proposed by Mohr *et al.* [19]. The authors use the projections of five scene reference points and the epipolar geometry (the "Essential" Matrix of [16] which is found by matching eight points) to determine the projections of the various stages of the construction needed to determine the three cross ratios for each point. The construction is elaborate and instead the authors propose and implement a direct non-linear algorithm for recovering the camera transformations be-

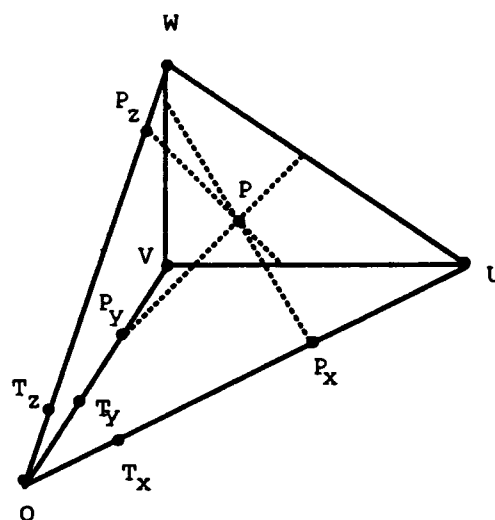


Figure 1: A non-homogeneous representation of space. The points O, U, V, W define the tetrahedron of reference. The point P_x is at the intersection of the plane PVW with the x -axis (the line OU). The point T_x is similarly constructed by replacing P with the unit point T (not shown in the drawing). The x coordinate of P is defined as the cross ratio of O, T_x, P_x, U (see [29], pp. 191).

tween the scene and the two views.

Faugeras [6] proposes a linear algorithm for recovering the camera transformations and the homogeneous coordinates. The projections of five scene reference points are used to determine each camera transformation matrix up to one unknown parameter (a camera transformation has 11 parameters and the correspondence between the reference points and their projections add five more unknowns, but produce 15 linear equations). The epipoles are then used as a sixth corresponding pair to fully determine (projectively speaking) the camera transformations. Once the camera transformations are recovered it becomes a simple matter to recover the homogeneous coordinates of any scene point whose projections in both views are known. Faugeras then considers the case of having four corresponding points instead of five. In that case the camera transformations are recovered up to four unknown parameters. Once these parameters are set (arbitrarily), then affine reconstruction becomes possible.

In our framework we do not recover the camera transformation matrices in order to achieve reconstruction. Instead we regard the camera's center as part of the projective reference frame making it necessary to use only four corresponding points coming from the scene. This still enables a projective reconstruction, and in addition to achieve an affine reconstruction in case the scene undergoes only affine transformations in space.

In the next section we derive the projective structure invariant and show how it can be computed given projections of four scene reference points (four corresponding points) and the corresponding epipoles. Section 4 describes the method by which 3D reconstruc-

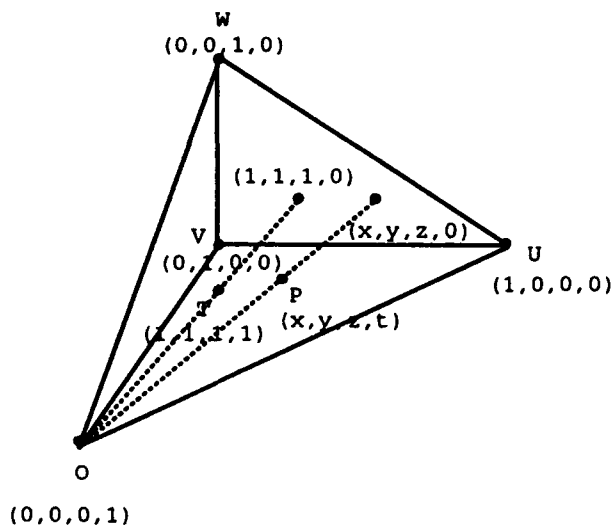


Figure 2: Homogeneous coordinates in space. If P is any point not on a face of the tetrahedron of reference, there exists four numbers x, y, z, t , all different from zero, such that the projections of P from the four vertices $(1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1)$ respectively onto their opposite faces are $(0, y, z, t), (x, 0, z, t), (x, y, 0, t), (x, y, z, 0)$ respectively (see [29], pp. 194-195).

tion is achieved given the recovered invariant. Section 5 describes two schemes for achieving re-projection, one using the invariant directly, and the other using the reconstructed structure. Section 6 briefly goes over two schemes for recovering the epipoles. Finally, Section 7 shows computer simulations intended to test the robustness of the schemes against noise in image correspondences.

3 The Projective Structure Invariant

Let the tetrahedron of reference consist of four scene points P_1, \dots, P_4 and let the fifth reference point be the camera's COP denoted by O . Let P be an arbitrary point of interest, and consider the ray from O to P . As illustrated in Figure 3, the ray OP intersects the two faces $P_1P_2P_3$ and $P_2P_3P_4$ at \tilde{P} and \hat{P} , respectively. We define our projective structure invariant as the cross ratio of P, \tilde{P}, \hat{P}, O , denoted by α_p :

$$\alpha_p = \langle P, \tilde{P}, \hat{P}, O \rangle = \frac{\tilde{P} - O}{\hat{P} - \tilde{P}} \cdot \frac{P - \hat{P}}{P - O},$$

where distances are measured along the ray OP . We will use α_p for reconstructing the homogeneous coordinates of P and for re-projecting P onto novel views, but first we describe the way α_p can be computed from image measurements alone.

In the first view all points along the ray OP project onto a single point, denoted by p , in the image plane. Because internal camera parameters are folded into the affine component of camera motion, we can assign $p = (x, y, 1)$ where (x, y) are the observed image coordinates

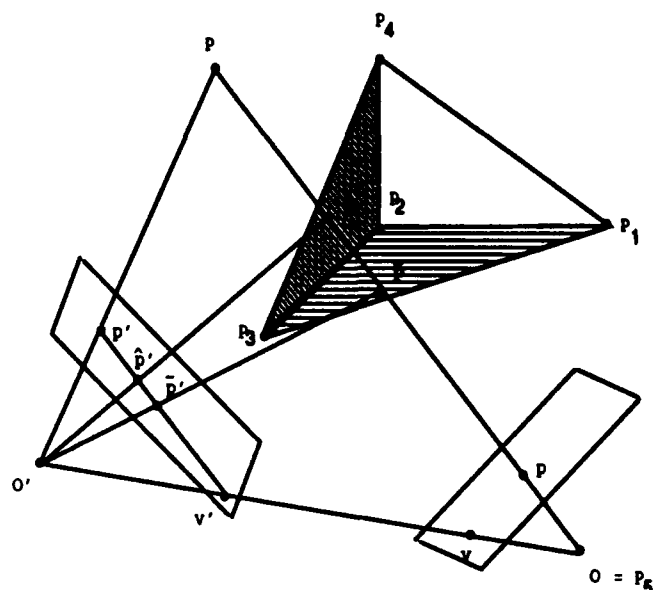


Figure 3: Projective structure of a scene point P is defined with respect to four reference points P_1, \dots, P_4 and the center of projection O of the first camera position. The camera's center serves as the unit point in the projective frame of reference instead of a fifth scene point. The cross ratio, denoted by α_p , of the four points P, \tilde{P}, \hat{P}, O uniquely fixes P with respect to the frame of reference. The cross ratio can be computed from the projections of P, \tilde{P}, \hat{P}, O onto the second image plane. The projection of O is the epipole v' which can be computed from eight corresponding points [6]; the other projections \tilde{p}', \hat{p}' can be recovered using the projections of the four reference points and the corresponding epipoles v, v' . Finally, since α_p is invariant it can be used for re-projection onto a third view and for reconstructing the projective structure of the scene.

with respect to some image origin (say the geometric center of the image plane). Consider next a second view of the scene. The points P, \tilde{P}, \hat{P}, O project onto generally distinct points denoted by $p', \tilde{p}', \hat{p}', v'$ which are also collinear. Because the two tetrads of points are projectively related, we have

$$\alpha_p = \langle P, \tilde{P}, \hat{P}, O \rangle = \langle p', \tilde{p}', \hat{p}', v' \rangle,$$

and therefore the structure invariant α_p can be computed from the projections onto the second view. The projection of O onto the second view is the epipole v' , and similarly the projection of O' (the COP of the second camera position) onto the first view defines the other epipole v , and therefore v and v' are corresponding points. We assume for now that the epipoles are known, and we will address the problem of finding them later (Section 6). The point p' is given to us (as we assume that correspondences between the two views has been established, as for example by [22, 23, 2]), and we can assign the coordinates $p' = (x', y', 1)$, where (x', y') are the observed image coordinates with respect to an arbitrary image origin. What is left is to recover the points \tilde{p}' and \hat{p}' .

In order to determine \tilde{p}' and \tilde{p} we must recover the projective transformations due to the two faces $P_1P_2P_3$ and $P_2P_3P_4$, respectively. This can be done by identifying four coplanar points on each of the two faces, but instead we can make use of the epipoles again. For example, we can use the projections of P_1, P_2, P_3 onto both views and the corresponding epipoles to uniquely recover the 2D projective transformation A , that when applied to p will produce \tilde{p}' , up to a scale factor. This is expressed in the following proposition:

Proposition 1 *A projective transformation, A , which is determined from three arbitrary, non-collinear, corresponding points and the corresponding epipoles, is a projective transformation of the plane passing through the three object points which project onto the corresponding image points. The transformation A is an induced epipolar transformation, i.e., the ray Ap intersects the epipolar line $p'v'$ for any arbitrary image point p and its corresponding point p' .*

Proof: Let $p_j \longleftrightarrow p'_j$, $j = 1, 2, 3$, be three arbitrary corresponding points, and let v and v' denote the two epipoles. First note that the four points p_j and v and the corresponding points p'_j , v' are the projections of four coplanar points in the scene. The reason is that the plane defined by the three object points P_1, P_2, P_3 intersects the line OO' connecting the two centers of projection, at a point — regular or ideal. That point projects onto both epipoles. The transformation A , therefore, is a projective transformation of the plane $P_1P_2P_3$. Note that A is uniquely determined provided that no three of the four points are collinear.

Let $\mu\tilde{p}' = Ap$ for some arbitrary point p . Because lines are projective invariants, any point along the epipolar line pv must project onto the epipolar line $p'v'$. Hence, A is an induced epipolar transformation. \square

Given the epipoles, therefore, we need just three points to determine the correspondences of all other points coplanar with the plane passing through the three corresponding object points. The transformation (collineation) A of the face $P_1P_2P_3$ is determined from the following equations:

$$\begin{aligned} Ap_j &= \rho_j p'_j, & j &= 1, 2, 3 \\ Av &= \rho v', \end{aligned}$$

where ρ, ρ_j are unknown scalars, and $A_{3,3} = 1$. One can eliminate ρ, ρ_j from the equations and solve for the matrix A from the three corresponding points and the corresponding epipoles. This leads to a linear system of eight equations (for more details see appendices in [20, 23]). Similarly, we can solve for the matrix E accounting for the projection of the face $P_2P_3P_4$ from the equations below:

$$\begin{aligned} Ep_j &= \mu_j p'_j, & j &= 2, 3, 4 \\ Ev &= \mu v'. \end{aligned}$$

If we set $\tilde{p}' = Ap$ and $\tilde{p} = Ep$ (note that \tilde{p}' and \tilde{p} are somewhere along the rays $O'\tilde{P}$ and $O'\tilde{P}$, respectively), then the cross ratio α_p can be computed using the linear combination of rays result known in projective geometry

([10], for example) as follows: we represent p' and \tilde{p}' as linear combinations of v' and \tilde{p}' :

$$\begin{aligned} \rho p' &= v' + k\tilde{p}' \\ \mu \tilde{p}' &= v' + k'\tilde{p}', \end{aligned}$$

then $\alpha_p = \frac{k}{k'}$ (note that ρ and k are fully determined, and so are μ and k'). Note that we have made use of the epipoles twice in our derivations. First, is because of having O as one of our reference points — this by definition brings the epipoles into the picture. Second, the epipoles were used in order to determine the image correspondences due to two faces of the tetrahedron of reference. Without the epipoles we would have needed an extra point on each face, hence loosing some generality because some of the reference points would have been coplanar. The computations for recovering α_p are simple and linear, and for convenience are summarized below:

- 1: Recover the transformation A that satisfies $\rho v' = Av$ and $\rho_j p'_j = Ap_j$, $j = 1, 2, 3$. Similarly, recover the transformation E that satisfies $\mu v' = Ev$ and $\mu_j p'_j = Ep_j$, $j = 2, 3, 4$.
- 2: Compute α_p as the cross ratio of p', Ap, Ep, v' , for all points p .

One can easily see how the projective invariant can be used to re-project the scene onto a third view. Simply perform Step 1 between the first and novel view (only four corresponding points and the corresponding epipoles are required). For any fifth point p , its corresponding point p'' in the third image can be found via α_p that has been recovered from the correspondence between p and p' (three points on the epipolar line and the cross ratio uniquely determine the fourth point p''). We will discuss re-projection and 3D reconstruction in more detail later, but before doing that it may be worthwhile to consider the situation of orthographic projection.

As mentioned previously, it is the property of the projective framework that orthographic projection becomes a particular case that does not require special treatment — this because the reference frame can map onto any configuration including the case where O is at infinity. Within the proposed geometric construction there are two points worth mentioning regarding the case of orthographic projection. First, the invariant α_p remains fixed under any projective transformation of the second image plane (the view on which α_p is computed). In particular the projection onto the second view can be orthographic (cross ratios are well defined for parallel rays as well). Second, consider the case when the first view is orthographic, i.e., O is at infinity. In this case α_p turns into an affine structure invariant:

$$\alpha_p = \langle P, \tilde{P}, \hat{P}, \infty \rangle = \frac{\tilde{P} - \hat{P}}{P - \tilde{P}}.$$

As a result, the projective invariant is defined and recovered under both orthographic and perspective projections. Therefore, in addition to enabling the use of uncalibrated cameras, we have the property (associated

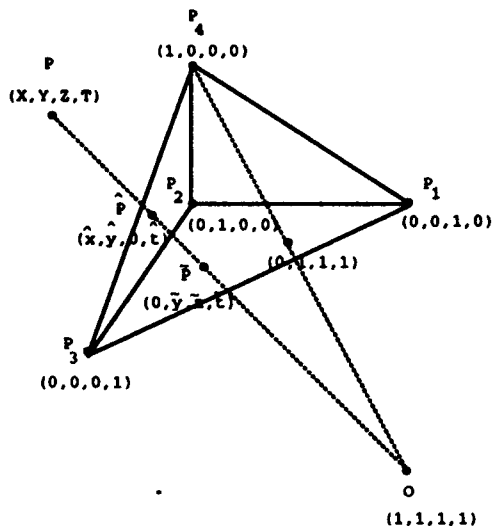


Figure 4: Reconstructing homogeneous coordinates of P (see text).

with the projective framework and not to the particular algorithm we proposed) that the size of field is no longer an issue as in a metric framework [1, 5].

We next show how to reconstruct the homogeneous coordinate representation of the scene given that we have recovered α_p . Taken together, the central result is that we can recover projective structure without recovering the camera transforms using only four corresponding points and the corresponding epipoles.

4 Reconstructing Homogeneous Coordinates

Given the invariant structure α_p we can easily reconstruct the homogeneous coordinates (X, Y, Z, T) of any fifth object point P (its actually a sixth point overall, but its the fifth object point). We first assign the standard projective coordinates to our frame of reference as follows: the coordinates $(1, 1, 1, 1)$ are assigned to O (the COP of the first camera position), then the coordinates $(0, 0, 1, 0)$, $(0, 1, 0, 0)$, $(0, 0, 0, 1)$ and $(1, 0, 0, 0)$ are assigned to the four reference points P_1, P_2, P_3 and P_4 , respectively (see Figure 4).

In this choice of coordinate system we have that $\tilde{P} = (0, \tilde{y}, \tilde{z}, \tilde{t})$ and $\hat{P} = (\hat{x}, \hat{y}, 0, \hat{t})$. Note also that the projection of P_4 onto the plane $P_1P_2P_3$ is the point with coordinates $(0, 1, 1, 1)$. In order to recover \tilde{P} we map the image plane onto the plane $P_1P_2P_3$ by solving for the projective transformation B that is determined by the four following correspondences. Let e_1, \dots, e_4 be the vectors $(0, 1, 0), (1, 0, 0), (0, 0, 1), (1, 1, 1)$. The correspondences $p_j \mapsto e_j, j = 1, \dots, 4$, fully determine the projective transformation B , i.e., $Bp_j = p_j e_j$. We can therefore set the coordinates of \tilde{P} :

$$\tilde{P} = \begin{pmatrix} 0 \\ Bp \end{pmatrix}.$$

In a similar fashion we can recover \hat{P} and with the knowl-

edge of α_p we can determine the coordinates of P . We can also do that in a simpler way without recovering \hat{P} , as follows. We know that

$$\mu \tilde{P} = O + s' \tilde{P},$$

$$\rho P = O + s \tilde{P},$$

and $\alpha_p = \frac{1}{\rho}$. Because the third coordinate of \tilde{P} is always zero, we have $s' = -\frac{1}{\rho}$. Thus,

$$P = O - \frac{\alpha_p}{\rho} \tilde{P}.$$

We have arrived to the following result:

Theorem 1 *In the case where the location of epipoles are known, then four corresponding points, coming from four non-coplanar points in space, are sufficient for computing the 3D homogeneous projective coordinates for all other points in space projecting onto corresponding points in both views. In the case the scene is undergoing an affine transformation in space, then the reconstructed scene is related to the true one by some unknown affine transformation.*

Note that the assignment of standard coordinates to the frame of reference is an arbitrary choice of representation and therefore, in the general case, the reconstructed structure is unique up to an unknown projective transformation of the scene. When the scene undergoes only affine transformations in space, then the COP can have fixed coordinates in space while allowing the remaining basis points P_1, \dots, P_4 to have any arbitrary representation in projective space. Because the COP is part of the reference frame, it is always assigned the same coordinates regardless of the viewing position from which we choose to reconstruct the scene. Therefore, the reconstructed scene, using the algorithm described above, will be unique up to an unknown affine transformation in space, and not a general projective transformation. For convenience one can projectively transform the reconstructed coordinates (X, Y, Z, T) to $(X, Y, Z, X + Y + Z + T)$ which ensures that the fourth coordinate is non-zero.

In comparison with Faugeras' [6] results, the bottom line is the same, i.e., with four corresponding points and the corresponding epipoles we can achieve 3D reconstruction of projective or affine space. The approach and the reconstruction algorithm are different, mainly because we go about the reconstruction process directly without first recovering the camera transformation matrices and instead recover first a geometric invariant α_p , which then can be used to reconstruct the homogeneous coordinates. Faugeras goes first through full reconstruction of the camera transformations using five corresponding points and the corresponding epipoles. In the case of four corresponding points (and the corresponding epipoles), Faugeras shows that the camera transformation can be recovered up to four unknown parameters. Once these parameters are set (arbitrarily) then reconstruction follows directly, and if one uses the same setting of the four parameters when reconstructing the scene from different view-points, then the reconstructions are only an affine transformation away from each other. In

our case, instead of fixing four parameters in the camera transformation from the scene to the first view, we fix the coordinates of the COP by having it being part of the reference frame.

We next discuss the use of these results (the projective invariant or the reconstructed scene) for obtaining re-projection onto a third view.

5 Achieving Re-projection

Considering the two views we worked with so far as "model" views of an object of interest, we can use the projective invariant α_p or the homogeneous coordinates to re-project the object onto any novel view given a small number of corresponding points across the three views.

First, consider the use of α_p to achieve re-projection. Assume we have four corresponding points across the three views $p_j \longleftrightarrow p'_j \longleftrightarrow p''_j$, $j = 1, \dots, 4$, and the epipoles v, v' between the two model views and u, u'' between the first model view and the novel view. From the correspondences $p_j \longleftrightarrow p'_j$, $j = 1, 2, 3$, and $u \longleftrightarrow u''$ we recover the collineation B , and similarly from the correspondences $p_j \longleftrightarrow p'_j$, $j = 2, 3, 4$, and $u \longleftrightarrow u''$ we recover the collineation D . Then, for any corresponding points $p \longleftrightarrow p'$, the third correspondence p'' can be recovered from the cross ratio α_p (computed from the two model views) and the three points Bp, Dp, u'' .

An alternative method is to first reconstruct the homogeneous coordinates of all points of interest from the two model views (by using four corresponding points and the corresponding epipoles). We then need only six corresponding points between the first model view and the novel view in order to recover the camera transformation matrix T from the scene onto the novel view:

$$\rho_j p'_j = T P_j \quad j = 1, \dots, 6.$$

Note that we have 11 unknowns for T and 6 more unknowns for ρ_j , but we have 18 linear equations. Then, for any point p for which we have recovered homogeneous coordinates of the corresponding scene point P , we can recover the projection of P onto the novel view by,

$$\rho p'' = T P.$$

This method, although less direct than the previous one does not require the epipoles between the first model view and the novel view (which requires eight corresponding points), and therefore achieves re-projection with fewer corresponding points with the novel view.

For completeness we review next two methods for recovering epipoles from point correspondences between two views. Both methods are linear — one requires correspondences coming from six points, four of which are assumed to be coplanar, and the second method requires eight general correspondences.

6 Recovering the Epipoles

The problem of recovering the epipoles is well known and several approaches have been suggested in the past [17, 21, 15, 8, 12, 7].

In general, the epipoles can be recovered from six points [15] (four of which are assumed to be coplanar),

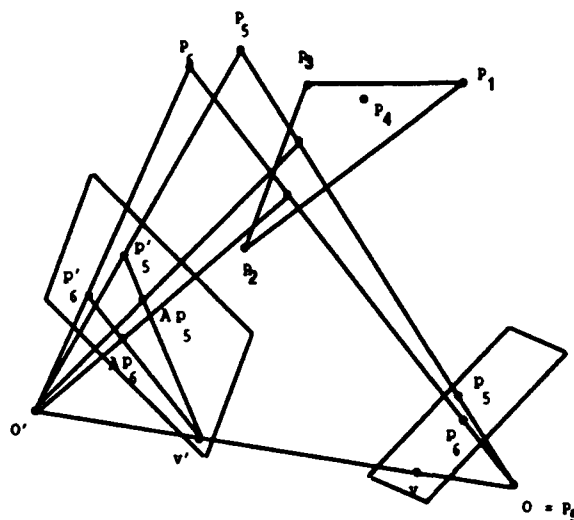


Figure 5: The geometry of locating the left epipole using two points out of the reference plane.

seven points (non-linear algorithm, see [8]), or eight points [6]. The basic idea behind the six point method is that the ray connecting the COP of the first camera position O and any object point P projects onto an epipolar line in the second image, and therefore the epipole can be found by intersecting two epipolar lines (see Figure 5). Given six points P_1, \dots, P_6 where P_1, \dots, P_4 are coplanar and P_5, P_6 are out of that plane, first recover the projective transformation A that satisfies $\rho_j p_j = A p_j$, $j = 1, \dots, 4$, then the epipoles v' and v are obtained as follows:

$$\begin{aligned} v' &= (p'_5 \times A p_5) \times (p'_6 \times A p_6), \\ v &= (p_5 \times A^{-1} p'_5) \times (p_6 \times A^{-1} p'_6). \end{aligned}$$

Note that the epipoles are represented as rays with respect to the camera centers, and therefore the case of parallel epipolar lines leads to a ray parallel to the image plane (third coordinate vanishes).

The basic idea behind the eight point method [6] is that since epipolar lines in both images are projectively related, then the epipolar geometry may be represented as a 2D correlation matrix. Let F be an epipolar transformation, i.e., $F l = \mu l'$, where $l = v \times p$ and $l' = v' \times p'$ are corresponding epipolar lines. We can rewrite the projective relation of epipolar lines using the matrix form of cross-products:

$$F(v \times p) = F[v]p = \rho l',$$

where $[v]$ is a skew symmetric matrix (and hence has rank 2). From the point/line incidence property we have that $p' \cdot l' = 0$ and therefore, $p'^T F[v]p = 0$, or $p'^T H p = 0$ where $H = F[v]$. The matrix H is a 2D correlation (i.e., maps points onto lines) and is also known as the "essential" matrix introduced by [16], and is of rank 2. One can recover H (up to a scale factor) directly from eight corresponding points, or by using a principle components approach if more than eight points are available. Finally, it is easy to see that $H v = 0$, and therefore the

epipole v can be uniquely recovered (up to a scale factor). Note that the determinant of the first principle minor of H vanishes in the case where v is an ideal point, i.e., $h_{11}h_{22} - h_{12}h_{21} = 0$. In that case, the x, y components of v can be recovered (up to a scale factor) from the third row of H .

7 Computer Simulation

We ran computer simulations to test the robustness of the re-projection method under various types of noise. Instead of measuring the error due to reconstruction we measured the errors due to re-projection onto a third view. The assumption being that the performance of the system (reconstruction and re-projection) largely depends on the quality of α_p , so we may as well observe noise effects on re-projection. We tested the system using both schemes for recovering the epipoles. In general, the 8-point scheme is significantly more sensitive to noise, and in practice additional corresponding points are required to achieve reasonable recovery of the epipoles. The experiments we describe below use the 6-point scheme for recovering the epipoles. Because the 6-point scheme requires that four of the corresponding points be projected from four coplanar points in space, it is of special interest to see how the method behaves under conditions that violate this assumption, and under noise conditions in general.

The object we used for the experiment consists of 26 points in space arranged in the following manner: 14 points are on a plane (reference plane) ortho-parallel to the image plane, and 12 points are out of the reference plane. The reference plane is located two focal lengths away from the center of projection (focal length is set to 50 units). The depth of out-of-plane points varies randomly between 10 to 25 units away from the reference plane. The x, y coordinates of all points, except the points P_1, \dots, P_6 , vary randomly between 0 — 240. The points P_1, \dots, P_6 have x, y coordinates that place these points all around the object (clustering these points together will inevitably contribute to instability).

We applied the following camera motion: The first view is simply a perspective projection of the object. The second view is a result of rotating the object around the point (128, 128, 100) with an axis of rotation described by the unit vector (0.14, 0.7, 0.7) by an angle of 29 degrees, followed by a perspective projection (note that rotation about a point in space is equivalent to rotation about the center of projection followed by translation). The third (novel) view is constructed in a similar manner with a rotation around the unit vector (0.7, 0.7, 0.14) by an angle of 17 degrees.

We conducted three types of experiments. The first experiment tested the stability under the situation where P_1, \dots, P_6 are non-coplanar object points. The second experiment tested stability under random noise added to all image points in all views, and the third experiment tested stability under the situation that less noise is added to the six points, than to other points.

7.1 Testing Deviation from Coplanarity

In this experiment we investigated the effect of translating P_1 along the optical axis (of the first camera position) from its initial position on the reference plane ($z = 100$) to the farthest depth position ($z = 125$), in increments of one unit at a time. The experiment was conducted using several objects of the type described above (the six points were fixed, the remaining points were assigned random positions in space in different trials), undergoing the same motion described above. The effect of depth translation to the level $z = 125$ on the location of p_1 is a shift of 0.93 pixels, on p'_1 is 1.58 pixels, and on the location of p''_1 is 3.26 pixels. Depth translation is therefore equivalent to perturbing the location of the projections of P_1 by various degrees (depending on the 3D motion parameters).

Figure 6 shows the average pixel error in re-projection over the entire range of depth translation. The average pixel error was measured as the average of deviations from the re-projected point to the actual location of the corresponding point in the novel view, taken over all points. Figure 6 also displays the result of re-projection for the case where P_1 is at $z = 125$. The average error is 1.31, and the maximal error (the point with the most deviation) is 7.1 pixels. The alignment between the re-projected image and the novel image is, for the most part, fairly accurate.

7.2 Situation of Random Noise to all Image Locations

We next add random noise to all image points in all three views (P_1 is set back to the reference plane). This experiment was done repeatedly over various degrees of noise and over several objects. The results shown here have noise levels between 0–1 pixels randomly added to the x and y coordinates separately. The maximal perturbation is therefore $\sqrt{2}$, and because the direction of perturbation is random, the maximal error in relative location is double, i.e., 2.8 pixels. Figure 7 shows the average pixel errors over 10 trials (one particular object, the same camera motion as before). The average error fluctuates around 1.6 pixels. Also shown is the result of re-projection on a typical trial with average error of 1.05 pixels, and maximal error of 5.41 pixels. The match between the re-projected image and the novel image is relatively good considering the amount of noise added.

7.3 Random Noise Case 2

A more realistic situation occurs when the magnitude of noise associated with the six points used for setting the construction (epipoles and projections of the tetrahedron of reference) is much lower than the noise associated with other points, for the reason that we are interested in tracking points of interest that are often associated with distinct intensity structure (such as the tip of the eye in a picture of a face). Correlation methods, for instance, are known to perform much better on such locations, than on areas having smooth intensity change, or areas where the change in intensity is one-dimensional. We therefore applied a level of 0–0.3 perturbation to the x and y coordinates of the six points, and a level of 0–1 to all other

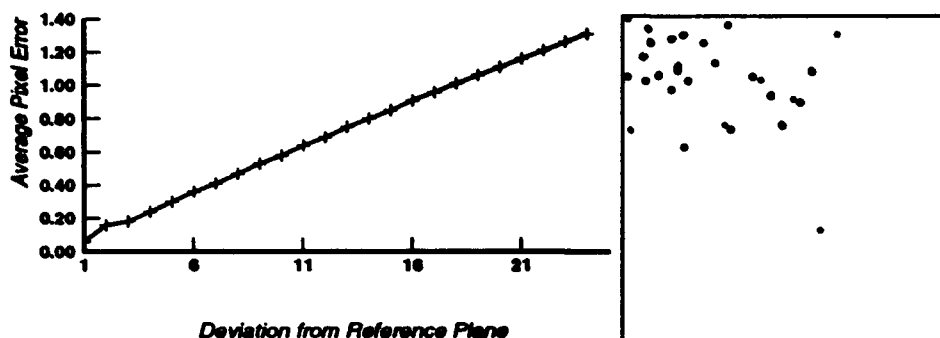


Figure 6: Deviation from coplanarity: average pixel error due to translation of P_1 along the optical axis from $z = 100$ to $z = 125$, by increments of one unit. The result of re-projection (overlay of re-projected image and novel image) for the case $z = 125$. The average error is 1.31 and the maximal error is 7.1.

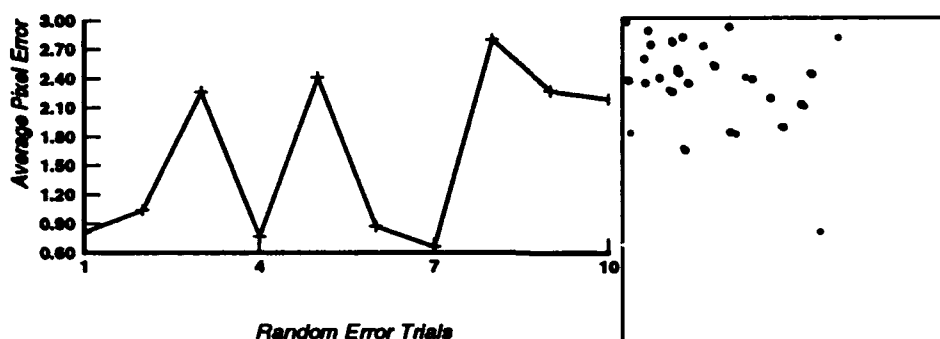


Figure 7: Random noise added to all image points, over all views, for 10 trials. Average pixel error fluctuates around 1.6 pixels. The result of re-projection on a typical trial with average error of 1.05 pixels, and maximal error of 5.41 pixels.

points (as before). The results are shown in Figure 8. The average pixel error over 10 trials fluctuates around 0.5 pixels, and the re-projection shown for a typical trial (average error 0.52, maximal error 1.61) is in relatively good correspondence with the novel view. With larger perturbations at a range of 0–2, the algorithm behaves proportionally well, i.e., the average error over 10 trials is 1.37.

8 Summary

We have described new techniques for two related problems: the problem of recovering structure from point matches, and the problem of visual recognition via alignment (the problem of re-projection). Our approach was based on recovering a geometric projective invariant that can then be used for both purposes: reconstruction and re-projection.

The key distinct features of our approach is, first, the role played by the center of projection and the epipoles. Second, the approach is primarily geometrically motivated with the definition of a new invariant which then drives the applications of reconstruction and re-projection. Thirdly, shape reconstruction and re-projection are achieved without going through the computations of the camera transformation matrices (e.g., structure without motion). The overall fea-

tures of the approach (shared with [6, 19, 11]) is that the system treats orthographic and perspective projections alike, and internal camera parameters are folded into the projection matrices, thereby allowing for views to be taken by uncalibrated cameras.

The structure invariant was recovered from four point matches arising from the projections of four non-coplanar object points, and the epipoles. The epipoles played a double role: first, the corresponding epipoles served as the projection of a fifth point in space, thereby allowing us to have a projective frame of reference while observing only four point matches from the scene. Second, with the epipoles we could determine the projections of various faces of the tetrahedron of reference — a task that otherwise would have required observing point matches coming from four coplanar points on each face. We then described two applications for which the invariant can be used for. First, we have shown that with the invariant we can achieve projective or affine reconstruction of the scene. Second, re-projection onto a third view was shown possible using the invariant directly without going through an explicit reconstruction of projective structure.

Finally, the algorithms for reconstruction requires eight corresponding points, or six assuming four of them are coming from coplanar points in the scene. For re-projection, the result is that the more we recover about

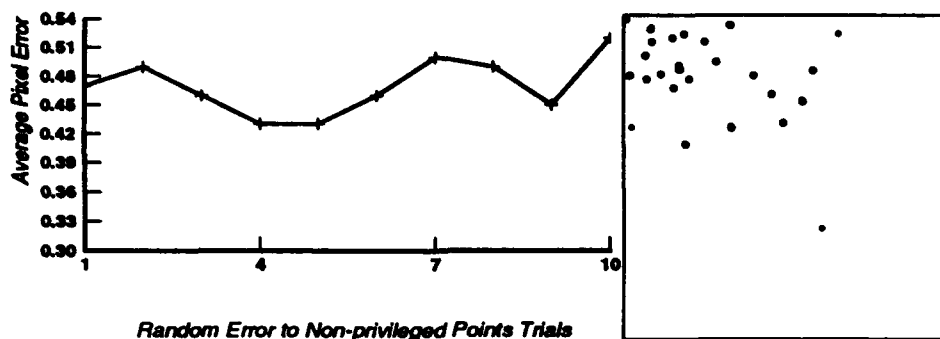


Figure 8: Random noise added to non-privileged image points, over all views, for 10 trials. Average pixel error fluctuates around 0.5 pixels. The result of re-projection on a typical trial with average error of 0.52 pixels, and maximal error of 1.61 pixels.

the scene and the camera transformation the less point matches are needed. We have seen that if projective structure is recovered, then only six point matches with the novel view are required for linear re-projection (via recovery of the camera transform matrix). If the projective invariant is used instead, then eight point matches are required.

Acknowledgments

I want to thank David Jacobs and Shimon Ullman for discussions and comments on this work. Thanks also to Roger Mohr for comments that improved the presentation of this paper.

References

- [1] G. Adiv. Inherent ambiguities in recovering 3D motion and structure from a noisy flow field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(5):477-489, 1989.
- [2] I.A. Bachelder and S. Ullman. Contour matching using local affine transformations. In *Proceedings Image Understanding Workshop*. Morgan Kaufmann, San Mateo, CA, 1992.
- [3] E.B. Barret, M.H. Brill, N.N. Haag, and P.M. Pyton. General methods for determining projective invariants in imagery. *Computer Vision, Graphics, and Image Processing*, 53:46-65, 1991.
- [4] T. Broida, S. Chandrashekhar, and R. Chellapa. recursive 3-d motion estimation from a monocular image sequence. *IEEE Transactions on Aerospace and Electronic Systems*, 26:639-656, 1990.
- [5] R. Dutta and M.A. Synder. Robustness of correspondence based structure from motion. In *Proceedings of the International Conference on Computer Vision*, pages 106-110, Osaka, Japan, December 1990.
- [6] O.D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Proceedings of the European Conference on Computer Vision*, pages 563-578, Santa Margherita Ligure, Italy, June 1992.
- [7] O.D. Faugeras, Q.T. Luong, and S.J. Maybank. Camera self calibration: Theory and experiments. In *Proceedings of the European Conference on Computer Vision*, pages 321-334, Santa Margherita Ligure, Italy, June 1992.
- [8] O.D. Faugeras and S. Maybank. Motion from point matches: Multiplicity of solutions. *International Journal of Computer Vision*, 4:225-246, 1990.
- [9] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381-395, 1981.
- [10] D. Gans. *Transformations and Geometries*. Appleton-Century-Crofts, New York, 1969.
- [11] R. Hartley, R. Gupta, and Tom Chang. Stereo from uncalibrated cameras. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 761-764, Champaign, IL., June 1992.
- [12] E.C. Hildreth. Recovering heading for visually-guided navigation. A.I. Memo No. 1297, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, June 1991.
- [13] D.P. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proceedings of the International Conference on Computer Vision*, pages 102-111, London, December 1987.
- [14] J.J. Koenderink and A.J. Van Doorn. Affine structure from motion. *Journal of the Optical Society of America*, 8:377-385, 1991.
- [15] C.H. Lee. Structure and motion from two perspective views via planar patch. In *Proceedings of the International Conference on Computer Vision*, pages 158-164, Tampa, FL, December 1988.
- [16] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133-135, 1981.
- [17] H.C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proceedings of the Royal Society of London B*, 208:385-397, 1980.

- [18] D.G. Lowe. *Perceptual organization and visual recognition*. Kluwer Academic Publishing, Hingham, MA, 1985.
- [19] R. Mohr, L. Quan, F. Veillon, and B. Boufama. Relative 3D reconstruction using multiple uncalibrated images. Technical Report RT 84-IMAG, LIFIA — IRIMAG, France, June 1992.
- [20] J. Mundy and A. Zisserman. Appendix — projective geometry for machine vision. In J. Mundy and A. Zisserman, editors, *Geometric invariances in computer vision*. MIT Press, Cambridge, 1992.
- [21] J.H. Rieger and D.T. Lawton. Processing differential image motion. *Journal of the Optical Society of America*, 2:354–360, 1985.
- [22] A. Shashua. Correspondence and affine shape from two orthographic views: Motion and Recognition. A.I. Memo No. 1327, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, December 1991.
- [23] A. Shashua. *Geometry and Photometry in 3D visual recognition*. PhD thesis, M.I.T Artificial Intelligence Laboratory, AI-TR-1401, November 1992.
- [24] A. Shashua. Projective structure from two uncalibrated images: structure from motion and recognition. A.I. Memo No. 1363, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, September 1992.
- [25] A. Shashua. Algebraic functions of image coordinates across three perspective/orthographic views. A.I. Memo No. 1405, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1993.
- [26] C. Tomasi. *shape and motion from image streams: a factorization method*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1991.
- [27] S. Ullman. Aligning pictorial descriptions: an approach to object recognition. *Cognition*, 32:193–254, 1989. Also: in MIT AI Memo 931, Dec. 1986.
- [28] S. Ullman and R. Basri. Recognition by linear combination of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13:992–1006, 1991. Also in M.I.T AI Memo 1052, 1989.
- [29] O. Veblen and J.W. Young. *Projective Geometry, Vol. 1*. Ginn and Company, 1910.

An Integrated Approach to Object Recognition

Noah S. Friedland

Computer Vision Laboratory, Center for Automation Research,
University of Maryland, College Park, MD 20742-3275

Abstract

A multilevel energy environment has been developed that simultaneously performs delineation, representation and classification of two-dimensional objects by using a global optimization technique. The energy environment supports a novel multipolar shape representation which allows the delineation and representation tasks to be viewed as a single operation. The delineator acts as a hypothesis generator for the multipolar representation, which uses minimum description length tests to determine whether to establish new polar centers. The polar representations at these centers are compared with a database of such representations in order to identify pieces of objects. This method is more robust than conventional multistaged approaches to object recognition because it incorporates all the information about the objects into a single decision process.

1 Introduction

This paper presents a novel approach to two-dimensional object recognition, which entails the solution of three sub-problems: delineation, representation and classification (DRC). Due to the difficulty of solving this problem in its totality, conventional object recognition systems address each of the sub-problems separately, and solve them sequentially. For example, a typical system might first extract the contour of a region (delineation); then polygonally approximate the contour (representation); and finally match pieces of the polygon to polygonal approximations of objects in a database (classification). Much work has been done on each of these sub-problems and the relevant literature is extensive. Attempts to concentrate on a particular stage always involve (often implicitly) assumptions regarding the quality of the available input information. These assumptions are often too strong and may lead to unpredictable errors in the recognition process.

A well-known theoretical framework describes the kinds of difficulties encountered by these conventional systems. Each of the steps in the DRC process can be thought of as a sequential stage in a communication system [8]. The output of each stage depends only on the

input to it from the previous stage. Important information may be lost through this communication process. For example, in the typical approach to DRC described above, the delineator "sees" only the input image and its performance is dependent on the extraction parameters. The edge information passed on to the representor is generally incomplete and may contain false alarms, thus potentially propagating errors of the first and second kinds. The representor creates its polygonal approximation of the contour based only on this edge map; this process can introduce its own errors into the system. These compounded errors are then carried on to the classification stage, where the final decisions of the recognition process are made.

Energy function (EF) based approaches are of interest in the DRC problem because of their ability to incorporate many different objective functions into a single cost to be minimized. These approaches include a paradigm called "snakes" first proposed by Kass, Witkin and Terzopoulos [10]. This is an active contour model approach which uses controlled continuity splines. The EF consists of a linear combination of three components which attract the snake to edges, lines and terminations. The first component attempts to minimize a cost relating to the image data—for example, to maximize edge strength or stereo disparity. The next component minimizes internal energy and has an effect on how flexible the snake is allowed to be. These components are integrated over the contour; hence, local influences are propagated globally throughout the contour. Other components allow external influences or volumetric information [2] to affect the delineation process. The optimization process is driven by an iterative solution of the Euler equations; it can be trapped at local minima and is therefore sensitive to initialization. The process is also computationally expensive [12]. Moreover, it does not provide a mechanism for incorporating knowledge about the expected region shapes into the delineation process; in other words, it does not integrate the processes of delineation and classification.

Another EF approach is typified by the minimum description length (MDL) algorithm due to Leclerc [11]. Here the function involves two components: a "language" component which represents chain code descriptions of region boundaries, and a "noise" component which measures the degree of variability of the pixel

intensities within the regions. The function is applied globally (i.e. to the entire image), starting with the partition in which each pixel is a separate region, and employs a graduated nonconvexity algorithm [1] to drive the description length to a near global minimum. This approach is computationally expensive and provides only a global mechanism (the non-convexity parameter) for guiding the search. Fua and Hanson [6, 7] presented an MDL-based DRC technique using a high-level component which incorporated chain code models for object shapes. Due to its use of a global approach their algorithm could not easily perform local description adjustments because these adjustments had little effect on the global cost.

Another possible framework for addressing the DRC problem has emerged from the work of Geman and Geman [9]. By pointing out the duality between Markov Random Fields (MRFs), which are defined by conditional spatial probabilities, and Gibbs distributions, which are determined by an EF, Geman and Geman provided a domain within which sub-processes could operate and communicate. Moreover, the resulting algorithms are naturally parallelizable, and also contain a natural mechanism for hypothesis generation, namely the Gibbs sampler.

Geman and Geman employed a MAP estimation paradigm (which, together with some assumptions about the noise model, resulted in the EF format) to perform image restoration by simulated annealing, using a two-level MRF. At the first level, MRF sites were pixel intensities with spatially homogeneous clique potentials; the second level introduced a line process which eliminated the cliques at suspected contour locations. The line process did not provide an easy way of incorporating information about arbitrary region shapes. Also, the computational cost of implementing such an MRF, even for a 64×64 image, was prohibitive.

A more efficient way of using the Gemans' approach was introduced in a paper by Friedland and Adam [4]. Here, a one-dimensional cyclic MRF was proposed, where the MRF sites were radii emanating from a given center. This had the effect of reducing the size of the MRF by at least two orders of magnitude. The paper presented an approach to determining cavity boundaries in echocardiograms using an EF whose components represented edge strength, contour smoothness, and cavity volume. These factors were considered simultaneously since they all resided in the same EF.

This paper presents an integrated approach to DRC based on EF minimization. Section 2 describes a DRC algorithm for what we call "compact" 2-D objects. A compact object is star-shaped, i.e. its entire contour is visible from an interior point; in addition, we require that the distances from this point to the contour of the object are not highly variable, so that the contour can be represented in polar form by a smoothly varying radius function. Compact object DRC is performed using a 1-D cyclic MRF which provides a framework for a polar object representation, and in which delineation and classification are performed by appropriate EF components. Thus local and global criteria are combined in the same

EF, which results in a single-stage recognition process. The optimization method used is simulated annealing, which provides the means of driving the energy to its lowest value, and hence leads to an MRF state with the highest probability.

Sections 3 and 4 extend the approach to "compound" objects, which are unions of small numbers of compact objects; here the contour visibility requirement is relaxed and the objects are allowed to have deep concavities and lobes. For this class of objects we use a "multipolar" representation, in which segments of the object's contour are represented by radius functions defined over sectors that emanate from a set of centers, as described in Section 3. As we shall see in Section 4, this allows us to perform DRC using a set of sector MRFs. It also overcomes the occlusion problems from which the single-polar representation suffers.

2 Compact Object DRC

2.1 The 1-D Cyclic MRF

Let $\mathbf{R} = (r_1, \dots, r_n)$ be a vector of discrete random variables r_i which represent radii emanating from a given center, $1 \leq i \leq n$, and let $\omega = (\omega_1, \dots, \omega_n)$, a vector of radius lengths, define a possible configuration ($r_1 = \omega_1, r_2 = \omega_2, \dots, r_n = \omega_n$) of \mathbf{R} ; the set Ω of such ω 's is the MRF's sample space. A 1DCMRF is then defined by

$$P(\mathbf{R} = \omega) > 0 \quad \forall \omega \in \Omega \quad (1)$$

$$P(r_i = \omega_i | r_j = \omega_j; j \neq i) = P(r_i = \omega_i | r_j = \omega_j; j \in N_i) \quad (2)$$

$$\forall i \in (1, 2, \dots, n); \forall \omega \in \Omega,$$

where N_i is a neighborhood of r_i . For simplicity, we take N_i to be $(i-1, i+1)$ and we use equispaced radii at angular intervals of $\Delta\theta = \frac{2\pi}{n}$. Since we are dealing with a closed contour, $r_{n+1} = r_1$ and the neighborhoods are defined modulo n . We also assume that the ω_i take on discrete values in a bounded range $[1, \omega^{\text{ext}}]$.

Figure 1 illustrates a 1DCMRF, its sites (radii), its sample space (radius values), its center location and its neighborhood system. Note that an MRF configuration $\mathbf{R} = \omega$ defines the polar representation of a contour relative to the given center.

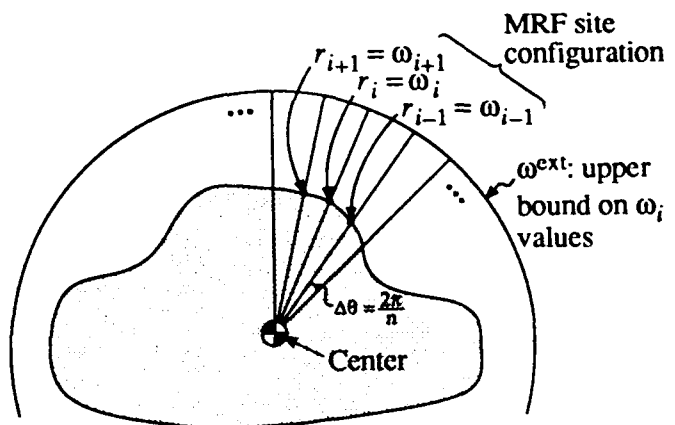


Figure 1: The 1-D cyclic Markov random field.

The 1DCMRF is initialized by choosing a center, which is assumed to lie somewhere inside the object's contour. (The problem of object detection is not addressed here; if an object has been detected but its location is known only approximately, it may be necessary to try a set of initial centers in order to insure that at least one of them lies inside the object.) Each of the n radii emanating from the center is assigned a randomly chosen initial value in the range $[1, \omega^{\text{ext}}]$.

Our implementation of the 1DCMRF allows the center to shift. At given steps in the optimization process we calculate the centroid of the current 1DCMRF configuration ω . This centroid becomes the new center. Using this new center the 1DCMRF is regenerated; ω is transformed into ω' defined relative to the new center. This ability to shift the center eliminates the need to choose a "good" center initially. As we shall see (Figure 2), the shifting process tends to eventually shift the center to a position close to the object's centroid. As a result, the polar representations of similar objects tend to be similar; thus it is meaningful to use these representations for classification purposes.

2.2 The Energy Function

Creating an EF framework for compact object DRC involves defining energy components to perform the appropriate subtasks. Our EF consists of two parts: a Low Level (LL), consisting of locally computed quantities that are used to delineate the object's contour; and a High Level (HL), which matches the polar representation of the contour to a database of polar contour models to perform classification. Symbolically,

$$E(\omega) \triangleq W_1 \times E_{LL}(\omega) + W_2 \times E_{HL}(\omega) \quad (3)$$

where W_1, W_2 are weights and E_{LL}, E_{HL} are the LL and HL energy components, respectively. The weight W_2 in Equation (3) is allowed to vary in the course of the optimization; its value depends on the difference between the current polar representation and its best-matching model in the database, as described in Section 2.2.2.

2.2.1 The Low Level

The LL component of the EF favors a delineation of the object that maximizes contour smoothness and edge sharpness. These properties depend on very localized regions along the contour. Edge sharpness is measured along each radius individually, while contour smoothness is measured by comparing the values of each radius r_i and its neighbors r_{i-1} and r_{i+1} .

The contour (non)smoothness measure is

$$E_{\text{smoothness}}(\omega) \triangleq \sum_{i=1}^n \frac{|\omega_i - .5 \times (\omega_{i-1} + \omega_{i+1})|}{\omega^{\text{ext}}}, \quad (4)$$

where $\omega_{i-1}, \omega_i, \omega_{i+1}$ are consecutive radius values. The edge sharpness measure along each radius is a difference of average gray levels; thus

$$E_{\text{step}}(\omega) \triangleq \frac{1}{M} \sum_{i=1}^n \left(\sum_{k=1}^M g(\omega_i + k) - \sum_{k=0}^{M-1} g(\omega_i - k) \right), \quad (5)$$

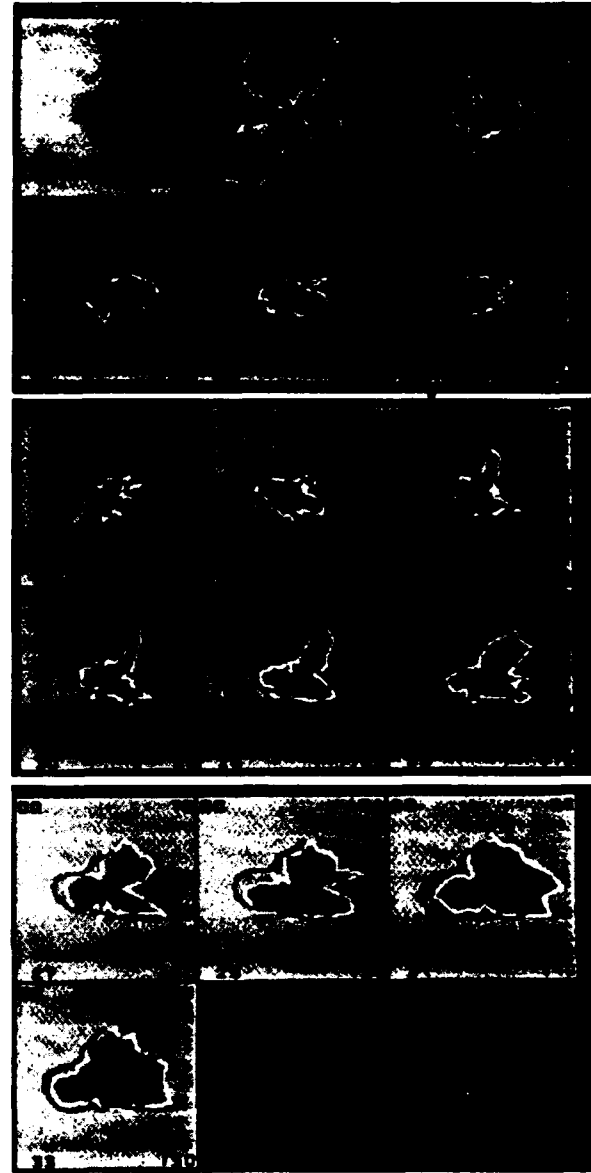


Figure 2: An example of our approach to compact object DRC.

where $g(\cdot)$ is the gray level at a given point. These two measures are combined linearly into the LL component of the EF:

$$E_{LL} \triangleq \alpha_1 \times E_{\text{smoothness}}(\omega) + \alpha_2 \times \left(1 - \frac{|E_{\text{step}}(\omega)|}{\text{MAXGRAY}} \right), \quad (6)$$

where α_1, α_2 are weights and MAXGRAY is the gray level range in the image.

The LL can also be used to introduce information about the contrast between the object and the background. If we have prior knowledge about what the contrast of the edges should be, edges with improper contrasts can be ignored in Equation (5), allowing the LL to consider relevant edges only.

The LL component of the EF is assigned a constant weight W_1 throughout the optimization. Its initial role is to construct an initial approximation of the object's

contour; later, when the HL component acquires more weight (see Section 2.2.2.), the LL component insures that the HL does not "hallucinate" matches that are not corroborated by the image data.

2.2.2 The High Level

The HL component is responsible for object classification. Its contribution to the optimization process consists of two parts: a fast process, which actively participates in the optimization at every iteration, and a slow process which, at regular iteration intervals, selects an object model that best matches the current MRF configuration (i.e., the current polar contour representation) and adjusts the value of the weight W_2 . Specifically,

- At every k^{th} step (for some preselected k) of the iterative optimization process, we compare the current MRF configuration $R = \omega$ with a database of polar contour models $S^{\text{obj},\theta}$, where obj is an object and θ is the object's orientation. The models are stored in normalized form, i.e. each of them is divided by the mean of its radius values, and are rescaled to $\bar{\omega}$, the mean of ω , for matching purposes.

- In the matching process, the match error is defined by

$$Er^{\text{obj},\theta} \triangleq \frac{1}{n} \sum_{i=1}^n \frac{|\omega_i - \bar{\omega} \times s_i^{\text{obj},\theta}|}{\bar{\omega} \times s_i^{\text{obj},\theta}}, \quad (7)$$

where $s_i^{\text{obj},\theta}$ is the i^{th} radius value of the model $S^{\text{obj},\theta}$. The process selects the model $T^{\text{obj},\theta}$ that has the lowest Er value. This error value can be used to define a halting criterion by comparing it to a cutoff threshold.

The weight W_2 assigned to E_{HL} is inversely dependent on the lowest Er value:

$$W_2 \triangleq \alpha_3 \times \max_{\text{obj},\theta} (1 - Er^{\text{obj},\theta}). \quad (8)$$

The updated HL component is the sum of the quantities

$$E_{HL}(\omega_i) \triangleq \frac{|\omega_i - \bar{\omega} \times t_i^{\text{obj},\theta}|}{\omega_{\text{ext}}}. \quad (9)$$

The weight W_2 controls the influence of the selected object model on the optimization. The smaller the error, the larger W_2 becomes relative to W_1 . Thus the HL component acquires more weight as its confidence in its match increases. Since all the models in the database are compact, any of them helps the LL attain contour smoothness in the initial stages of the optimization. As the optimization approaches the global minimum of the energy, and the MRF configuration approaches one of the models in the database, W_2 grows. As a result the HL component becomes more dominant and speeds up the final convergence of the optimization.

Note that the HL is an active participant in the EF, not merely a postprocess. This gives the system top-down qualities, a very important point. Also, by providing a halting criterion, the HL allows a result of "don't know" to be obtained.

2.3 The Compact Object DRC Algorithm

The algorithm is initialized by choosing a center and establishing a 1DCMRF R by choosing n radii emanating from this center at angular intervals of $2\pi/n$. A random initial guess configuration $R = \omega_0$ is assigned to the 1DCMRF so that an initial Gibbs density, which corresponds to the probability density of each site $r_i \in R$, can be calculated (see [4]). Then the optimization process is allowed to begin. At every k^{th} iteration of the process, the center is shifted to the centroid of ω . A model which best matches the resulting configuration is selected from the database. This model is subsequently used in the HL component of the EF. The Gibbs densities are also re-determined. This process continues until either a closely matching model has been found, the halting criterion has been satisfied, or an upper bound on the number of iterations has been reached.

It is important to stress that the process uses the same EF system throughout, though the EF itself undergoes changes as the process proceeds.

2.4 Experimental Results

In this section we demonstrate some of the characteristics of the compact DRC algorithm through an example, using an image of a tank obtained by a forward looking infra-red (FLIR) sensor. Figure 2 shows the MRF configuration at every tenth iteration. The initial image window (85 by 85 pixels) is shown on the upper left; the location of the center is marked with a small +. Note that throughout the process, the + remains in the center of the window; as the center shifts, the window shifts with it.

The images have had their gray scales compressed from [0,255] to [60,195] for display purposes. The white contour (gray level 255) superimposed on each image shows the current MRF configuration. The black contour (gray level 0) shows the current best matching model, scaled by $\bar{\omega}$ and rotated by θ . The number in the top left corner of each image designates this object, and the number in the upper right corner represents its degree of match, $1 - Er$. The two lower numbers are the contrast between the object and the background (left) and the iteration number (right).

We see that in the first 70 iterations the algorithm thinks it has detected a tank (model no. 1 or 2) in the lower left part of the target. At iteration 80, the LL begins to express its "doubts" about the validity of this erroneous target identification. As the MRF begins to acquire the correct target (no. 0), center shifting allows the MRF to generate configurations which increasingly resemble the correct result. This was possible because simulated annealing allowed the process to escape the local minimum in which it was initially trapped.

More details about our MRF approach to compact object DRC, and many additional examples, can be found in [5]. It should be pointed out that major occlusions cannot be handled by this approach, because they introduce major distortions in the polar representation, which interfere with finding correct model matches. In the following sections we will introduce a more flexible approach.

3 The Multipolar Representation

3.1 Introduction

In this section a novel multipolar shape representation scheme, MPR, is presented. MPR generalizes the polar representation (PR) used in Section 2, which consists of a center (x_0, y_0) and n radii spaced at angular intervals of $2\pi/n$, to a form which allows a shape to be represented by many "centers" (x_i, y_i) , $i \in (1, 2, \dots, N)$, each with n_i radii, spaced over an angular sector at angular intervals of $2\pi/n$, which define a polar representation of a segment of the shape's contour.

MPR, because of its multiple centers, is far less sensitive to occlusion than PR because matches to unoccluded contour segments can still be found. At the same time MPR shares many of PR's strengths. Its shape descriptions are relatively concise and are invariant to scaling, translation and rotation. Also, like PR, MPR is highly compatible with MRF environments.

Several stages are involved in creating an MPR. Initially, a PR of the shape is created. Next, contour curvature extrema are detected. Due to the discrete nature of the representation an approximating "sag function" is used to estimate the curvature, and extrema of this function are found (see Section 3.2). Segments of the contour that contain extrema define candidate MPR centers. Internal centers, i.e. centers which are inside the contour, are defined for contour segments that contain curvature maxima, and external centers, lying outside the contour, for segments that contain minima (i.e., negative maxima).

To insure a compact representation a candidate MPR undergoes a minimum description length (MDL) test, and is accepted only if it is more compact than the original PR. This process is then repeated for each of the centers in the MPR until no further creation of new centers is accepted. To avoid fragmentation, a lower bound is imposed on the number of radii associated with each center.

3.2 Contour Segmentation

A simple function which we call the "sag function" is used as an estimate of the curvature of the contour. The sag at r_i is calculated using $r_i(x, y)$, $r_{i-1}(x, y)$ and $r_{i+1}(x, y)$, the Cartesian coordinates of the endpoints of radii r_i , r_{i-1} and r_{i+1} , where r_{i-1} and r_{i+1} are the left and right neighbor radii of r_i . (If r_i is the first or last radius of its contour segment, one of its neighbors is taken to be the last or first radius of the adjacent segment.) Let $d_2(r_i)$ be the length of the chord $r_{i-1}(x, y)r_{i+1}(x, y)$, and let $d_1(r_i)$ be the length of the perpendicular from $r_i(x, y)$ to the chord (see Figure 3). Sag(r_i) is then defined as

$$\text{Sag}(r_i) = \begin{cases} \frac{d_1}{d_2} & \text{if } \left| \frac{d_1}{d_2} \right| > \epsilon \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

where ϵ is a small quantity used to eliminate spurious extrema along straight contour segments. They can also be eliminated by smoothing the contour before calculating the sag.

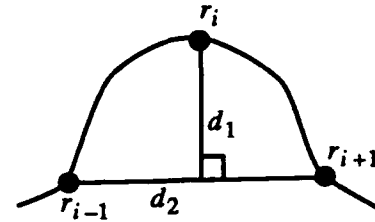


Figure 3: The sag function.

Although the sag function is not a true approximation to curvature (being scale invariant), it does have the following properties:

- For a straight line, where the curvature is zero, d_1 is also zero, so the sag is zero.
- At curvature extrema d_1 is maximized, while d_2 is minimized; hence the sag function has maxima at the same locations as the curvature, provided the radii are spaced closely enough to capture the contour's behavior.

After the sag function has been calculated at each radius r_i , its significant extrema are extracted using the following heuristics:

- A maximum/minimum location must have a positive/negative sag value.
- The nearest neighbors of a maximum/minimum must also have positive/negative or zero sag values.
- The next nearest neighbors must have sag values that have at most a given difference from the values at the nearest neighbors.

Once these extrema are identified the contour is partitioned into segments. A partition point r^* is chosen between each pair of consecutive extrema as follows: If the extrema are of opposite type, r^* is chosen at the location of the most significant sag zero-crossing. If they are of the same type, r^* is chosen at the location of the most significant instance of sag of the opposite type.

For each contour segment (r^* -extremum- r^*), we estimate the coordinates (x_{cg}, y_{cg}) of the center of a circular-arc fit to the segment. Note that if the extremum is positive (the segment is convex), the center will be internal to the contour, while if it is negative (a concave segment), the center will be external.

This method of choosing candidate centers has a number of benefits. A representation based on a partition of the object's contour into segments, each containing a significant curvature extremum, is quasi-invariant, i.e., it does not change greatly under modest aspect changes. Thus the MPR does not vary significantly if aspect changes are small. Furthermore, occlusion nearly always involves a situation where a convex part is missing; thus a representation which breaks the contour into convex and concave segments provides a natural framework for dealing with occlusion.

The contour segment is now represented in a polar coordinate system centered at (x_{cg}, y_{cg}) . The angular support of the segment relative to this center is tested for a systematic counterclockwise (for internal centers) or clockwise (for external centers) progression. A portion of

the segment that violates this condition is merged with a neighboring segment of opposite type. If such a neighbor does not exist, a new candidate center of opposite type (external or internal) is defined for that portion.

3.3 The MDL Criterion

The number of bits required to encode a given realization of an i.i.d. Gaussian random vector (x_1, \dots, x_m) is given by

$$\frac{m}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} \sum_{j=1}^m (x_j - \mu)^2 \quad (11)$$

where μ is the mean of the vector, σ^2 is its variance, and the units are bits.

Using Equation (11), we can compute the number of bits needed to describe a candidate center as follows:

- The coordinates of the center: 2×9 bits (assuming a 512×512 image).
- The directions of the left and right boundary radii: 2×5 bits (assuming 32 radii per center).
- The mean radius length μ : 6 bits (assuming that the range of radius values is 64).
- The number of bits needed to encode the variations from μ : $\frac{m}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} \sum_{j=1}^m (l_j - \mu)^2$ (assuming the center has m radii of lengths l_1, \dots, l_m which have i.i.d. Gaussian distributions around μ).

Therefore approximately

$$b_i = 34 + \frac{n_i}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} \sum_{j=1}^{n_i} (l_j - \mu_i)^2 \quad (12)$$

bits are needed for a center that has n_i radii with mean length μ_i . The variance σ^2 is assumed to be the same for all centers; it needs to be determined from observation.

The MDL test used to determine whether the candidate centers actually become new centers is then

$$\sum_{i=1}^K b_i < b_0 \quad (13)$$

where K is the number of candidate centers, b_i are these centers' bit values, and b_0 is the bit value for the original center.

3.4 Examples

In this section we present a few simple examples of MPRs. In the figures, angular regions of support are indicated by black lines that run from each contour segment's endpoints to the center associated with the segment. All the examples are initialized by a PR. If the MDL test is met the PR is partitioned (repeatedly, if necessary) into an MPR. The total description length in bits of the PR/MPR, as calculated for each center by Equation (12), appears in the lower righthand corner of each image, rounded to the nearest integer.

Figure 4 shows a four-lobed synthetic object. Here, although the PR adequately describes the contour, the MPR has a shorter description length than the PR. It also explicitly tells us that we have a four-lobed object



Figure 4: MPR example: four-lobed object.



Figure 5: MPR example: FLIR tank.

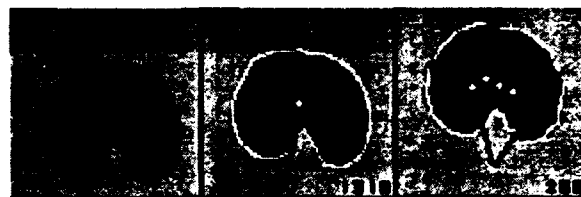


Figure 6: MPR example: leaf.

and provides descriptions of the lobes. Figures 5 and 6 show MPRs of two real objects: a tank taken from a FLIR image and a leaf.

These examples illustrate how the MPR representation provides a partition of an object into simple parts. The MDL test and the heuristic method of choosing significant extrema prevent excessive fractionation of the object. The result is an efficient and plausible representation. Numerous additional examples, and further details about MPR construction, can be found in [3].

4 Compound Object DRC

4.1 Introduction

The PR used for DRC in Section 2 changes significantly if a significant part of the object is missing, e.g. occluded, because the radii of the partial object are all measured from a different centroid. As a result the classification process will be ineffective even if the delineation is performed correctly. This problem applies to all global shape recognition techniques, i.e. techniques in which recognition is based on properties of the entire shape. Because of this, when dealing with potentially occluded objects, shape recognition techniques which examine segments of the contour are usually employed.

In this section we describe an MRF-based DRC system which makes use of the MPR described in Section 3. The MRF structure consists of a set of radial MRFs, defined over angular sectors emanating from a set of centers; the configurations of these MRFs define polar representations of segments of the object's contour. This MPR-

MRF structure can handle compound objects, parts of which may be occluded.

4.2 The MPR-MRF

The compound DRC process is initialized, as in Section 2, by choosing a single center (x_0, y_0) inside the object and constructing a 1DCMRF. The initial stage of the algorithm compares the configurations of this MRF with PR models in the database to see if a good match can be found. If not, the algorithm proceeds to the next level of MPR partitioning and attempts to find matches between the configurations of the sector MRFs and parts of MPR models, i.e. polar representations of contour segments of database objects. At each sector MRF for which a match is found (as defined by a convergence criterion), the optimization process is terminated and the sector's center is not considered for further MPR partitioning.

Formally, the MPR-MRF Ξ consists of a set of sectors $\xi \in \Xi$ emanating from centers (x_ξ, y_ξ) . Each sector ξ consists of radial sites $r_i^\xi \in R^\xi$. As in Section 2.1, each radial site is a discrete random variable with values in the range $[1, \omega^{\text{ext}}]$. The radii are spaced at regular angular intervals of $2\pi/n$, and the neighbors of radius i are $i-1$ and $i+1$. Note that these sector MRFs are no longer cyclic.

To allow the sectors to align themselves relative to contour segments and provide initialization-independent polar representations of the segments, the MPR-MRF allows the centers to shift, as in Section 2.1. Ideally, the angular supports of the centers should just "cover" the entire contour. However, when a center shifts, its angular support changes, possibly resulting in incomplete or overlapping coverage of the contour. The center shifting algorithm tests for this and initializes additional radial sites, or deletes redundant sites, as needed. If this results in a center having its number of radial sites reduced below the fragmentation threshold, that center is eliminated and its sites are assigned to nearby centers.

4.3 The Energy Function

The MPR-MRF is responsible for the representation portion of the DRC task, while the delineation and classification portions are the responsibility of the EF. As in Section 2.2, the EF is defined symbolically as

$$E(\omega) \triangleq \sum_{\xi} \{W_1^\xi \times E_{LL}^\xi(\omega) + W_2^\xi \times E_{HL}^\xi(\omega)\}, \quad (14)$$

where ξ is the index representing the MPR-MRF sectors; E_{LL}^ξ , E_{HL}^ξ are the LL and HL energy components for each sector; and W_1^ξ , W_2^ξ are weights associated with each sector.

4.3.1 The Low Level

The LL components of the EF enable the system to perform object delineation. Their definitions are similar to those in Equations (4-6) of Section 2.2.1.

4.3.2 The High Level

The role of the HL is to periodically classify the contour segments found by the LL delineator. Each sector

ξ finds the best match between its current MRF configuration and a compatible (see below) contour segment model (i.e., part of an MPR model) in the database. The HL periodically resets the weight W_2^ξ to reflect the quality of this best match. Sectors ξ for which the best match is very good have high W_2^ξ values, so that the HL has high weight in the optimization process, while centers with only poor matches have low W_2^ξ values, so that the LL dominates.

The set of compatible contour segment models for sector ξ is defined by

$$\text{models}_\xi \triangleq \{k : (|v^k - v^\xi| \leq \text{thr}_v) \wedge (\text{type}_k = \text{type}_\xi)\}, \quad (15)$$

where k is a segment model; type is internal or external; v is a unit vector in the direction of the middle radius of the sector (or segment model); and thr_v is a threshold which depends on how much orientational freedom is expected in the objects to be recognized.

The match between ξ and model k ($k \in \text{models}_\xi$) is defined as

$$m_{\xi,k} \triangleq 1 - \frac{1}{\bar{\omega}^\xi} \sqrt{\frac{1}{n_\xi} \sum_{i=1}^{n_\xi} (\omega_i^\xi - \bar{\omega}_i^k)^2} \quad \text{if } m_{\xi,k} \geq \text{thr}_m; \quad (16)$$

0 otherwise,

where n_ξ is the number of radii in ξ , $\bar{\omega}^\xi$ is their average length, and thr_m is a threshold below which the match is considered inconclusive. The $\bar{\omega}_i^k$ are angularly aligned with their ω_i^ξ counterparts by making the middle radius of k coincide with the middle radius of ξ or with one of its neighbor radii. Radii of k that fall outside sector ξ are ignored, while radii of ξ that fall outside k 's sector are regarded as having corresponding radii of k that have zero length.

Let model k^ξ give rise to the maximum (nonzero) value of $m_{\xi,k}$. The updated HL component is then defined by

$$E_{HL}^\xi(\omega_i) = \min \left[\frac{|\omega_i^\xi - \bar{\omega}^\xi \times \omega_i^{k^\xi}|}{\text{SIZE}}, 1 \right] \quad (17)$$

where SIZE is a threshold, and the i 's in the numerator are offset if necessary to allow for the angular alignment. The weight W_2^ξ assigned to this component is

$$W_2^\xi \triangleq \alpha_{HL} \max_k \{m_{\xi,k}\} \quad (18)$$

4.3.3 Object Classification

Let Ξ' be the set of sectors ξ for which best matching models k^ξ have been found, and let $n' \triangleq \sum_{\xi \in \Xi'} n_\xi$ be the total angular size of these sectors. We assign the weight $w_\xi \triangleq n_\xi/n'$ to each of these sectors, and zero weights to the remaining sectors. This weighting scheme gives higher weights to classified sectors that have larger angular supports, since these sectors contain more contour information.



Figure 7: The original objects.

The evidence for a database object is then defined as

$$\epsilon_{obj} \triangleq \sum_{\xi \in \Xi} w_{\xi} m_{k\xi} \max\{\tilde{m}_{k\xi, l}\}, \quad (19)$$

where the max is taken over all the contour segment models l that are parts of MPR models for obj in the database, and \tilde{m} is a match measure between pairs of models, defined as in Equation (17). Evidently, if we are actually dealing with a database object, the evidence for that object should be very high, and it should be relatively high even if the object is partly occluded, as long as significant segments of its contour are not occluded.

4.4 The Compound Object DRC Algorithm

The MPR-MRF DRC algorithm consists of two processes: a fast process, which performs simulated annealing optimization on all MPR-MRF sites, optimizing the EF defined by Equation (14), and two slow processes—one which detects matches, updates the HL weights, and updates center locations, and one which performs contour partitioning and new center creation.

The algorithm is initialized by establishing a single-center MPR-MRF $R^0 = \omega^0$ which is equivalent to a 1DCMRF. This 1DCMRF configuration is used to determine the initial Gibbs density. At this early stage the HL component performs matching only between the MRF configuration and the PR models in the database.

In general, let the set of MPR-MRF centers be Ξ . For all $\xi \in \Xi$, the fast process iteratively visits each radial site r_i^{ξ} of ξ and attempts to modify its configuration ω_i^{ξ} . This is done by generating a candidate configuration using the Gibbs sampler, evaluating its energy using Equation (14), and deciding whether to accept the transition.

At every 10th iteration of the fast process, the first slow process updates each center's location to the centroid of its current sector MRF configuration, finds the best matches for these configurations, updates the sectors' W_2 values, and recalculates the centers' Gibbs samplers. Sector MRFs whose match values meet a convergence criterion are declared "converged" and the fast process is no longer applied to them. The evidence for each database object is checked at the end of this stage, using Equation (19), and compared to a threshold to determine if an object match has been found.

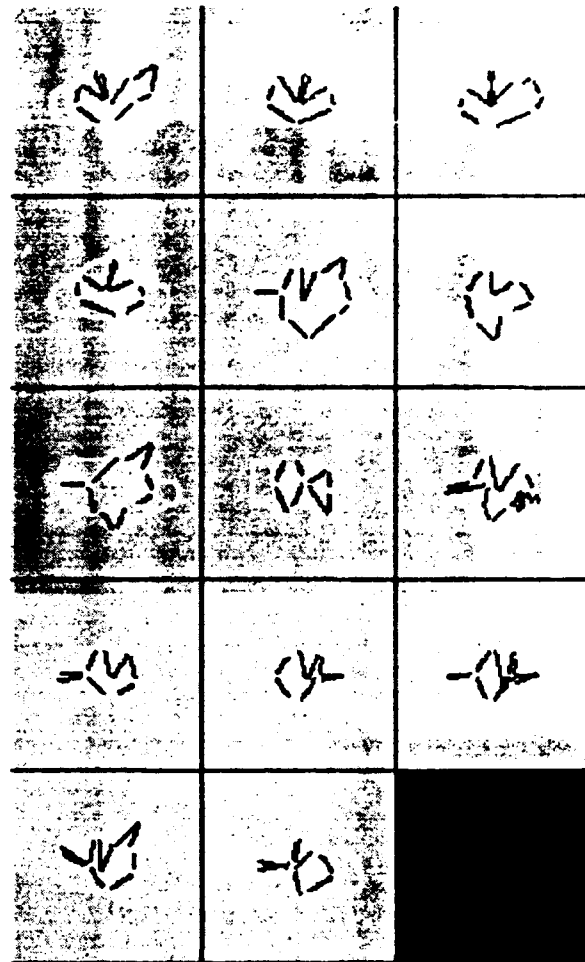


Figure 8: MPR-MRF database.

The second slow process partitions the contour segments of the non-converged sector MRFs and creates new centers, if the MDL criterion is satisfied; this is done every 100th iteration. In our experiments, we limited the total number of iterations to 290, i.e. the second slow process was applied only twice.

4.5 The Model Database

Our model database was constructed from a set of six "cut" objects on a uniform background which were sampled and digitized. The use of these artificial objects was necessary because it provided a domain in which distortion and occlusion could be controlled. The original objects are shown in Figure 7. They are (from left to right, top to bottom): tank1, tank2, truck1, truck2, apc1, and apc2.

An MPR-MRF delineation algorithm similar to the recognition algorithm was used to create PR and MPR representations of the objects starting from various initial center locations. This sometimes gave rise to more than one MPR for a given object, because the sparse set of radii sometimes failed to detect small features of the contour, resulting in variations in the contour segments. The database consisted of the PR and MPR representations of all the objects. Figure 8 shows all the MPRs in

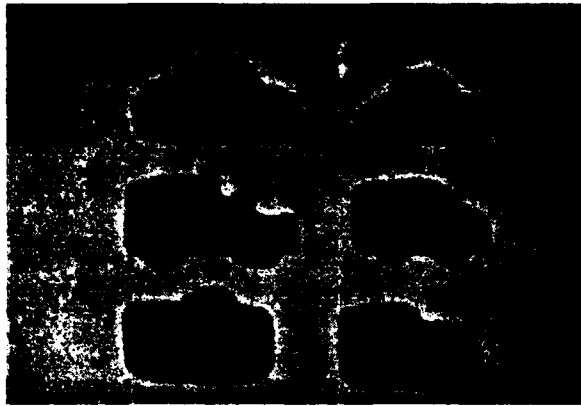


Figure 9: Distorted versions of the original objects.

the database. These MPRs all correspond to plausible descriptions of the objects. Due to the low resolution (30-50 pixels across an object), the possible number of variations is small.

4.6 Experimental Results

The compound DRC algorithm was tested on a set of real FLIR images of non-occluded targets resembling the database objects, as well as on images of both occluded and non-occluded distorted versions of the objects (examples of the distorted versions are shown in Figure 9). As an example, Table 1 and Figure 10 show results for a distorted version of *apc2*. The evidence column indicates the total evidence for each object. The numbered columns list the evidence contributions on a sector by sector basis. Figure 10 shows the optimization process for this example from initialization to termination (which occurred at 150 iterations because all sector MRFs converged). Iterations 00 through 90 show the development of the PR. At iteration 100, the black dots show the radius endpoints, and the white dots show the locations of significant sag extrema. The MPR constructed using these extrema reduced the description length from 1016 to 283 bits (see iteration 110) and was accepted. It has three internal centers (two of which co-incident) and one external center. During the succeeding 40 iterations, the sector MRFs all converged to configurations that were good matches to database models. When a sector MRF has converged to match part of an MPR model for a database object, the contour segment represented by this model is displayed in white, together with the number of the object (see iterations 140 and 150).

Table 1: MPR-MRF evidence for a distorted "*apc2*".

Index	Target	Evidence	1	2	3	4	5
12	<i>apc2</i> 1	.834	.445	.045	.000	.120	.224
9	<i>truck2</i> 3	.743	.382	.000	.206	.154	-
0	<i>tank1</i> 1	.082	.000	.082	.000	.000	.000

The next group of figures and tables shows results obtained for three non-occluded targets in real FLIR images, using the same model database. Each figure shows

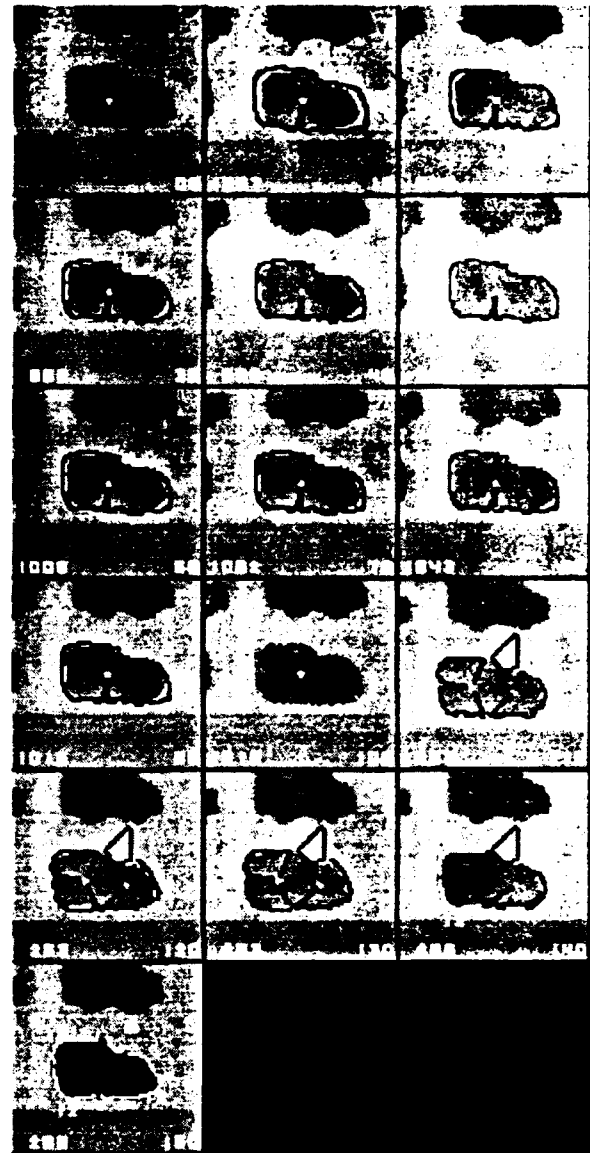


Figure 10: MPR-MRF results for a distorted "*apc2*".

only iterations 00, 90 (with the PR configuration and best matching PR model), and 290 (with the MPR configuration and best matching parts of MPR models). Figure 11 and Table 2 give results for a FLIR truck. Although the database does not include this truck, the process finds very high evidence for *truck1*, and little evidence for any other database object. The PR classification was *apc2*; however, once the MPR-MRF was initialized, this choice was discarded in favor of the more correct solution, *truck1*. The results are similar, though less clearcut, for a second truck, as shown in Figure 12 and Table 3. In Figure 13 and Table 4, results for a FLIR tank image are presented. *Tank1.1* and *tank2.2* are the contenders; the actual tank has a turret that is more similar to *tank2*'s, and a body that is slightly more similar to *tank1*'s.

Not surprisingly, the results for occluded objects tend to be more ambiguous, because the visible parts of such an object often resemble more than one database model.

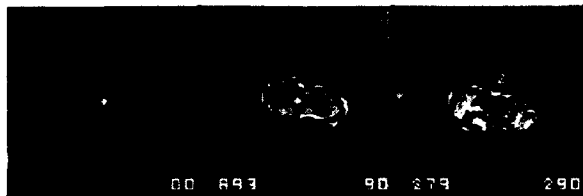


Figure 11: MPR-MRF results for the first truck in a FLIR image.



Figure 12: MPR-MRF results for the second truck in a FLIR image.



Figure 13: MPR-MRF results for a tank in a FLIR image.



Figure 14: MPR-MRF results for an occluded "truck2".

Figure 14 shows an example in which there is relatively little ambiguity (see Table 5), in spite of the fact that over half of the object is occluded. Many other examples, as well as further details about the MPR-MRF algorithm, can be found in [3].

5 Concluding remarks

The MPR-MRF system described here provides a powerful framework for integrated DRC. The EF environment allows versatility in integrating the DRC modules. Delineation and classification are handled by appropriate components of the EF, while the MPR provides the representation. The system poses the DRC task as an optimization problem and achieves a near-global optimum using simulated annealing. The experimental results demonstrate the ability of the integrated approach to identify a variety of objects under conditions of occlusion and distortion. Due to the low resolution, the number of MPR representations is limited, resulting in

Table 2: MPR-MRF evidence for the first truck in a FLIR image.

Index	Target	Evidence	1	2	3	4	5
5	truck1.2	.822	.471	.000	.351	.000	-
2	tank2.1	.061	.000	.061	.000	.000	.000

Table 3: MPR-MRF evidence for the second truck in a FLIR image.

Index	Target	Evidence	1	2	3	4	5	6
8	truck1.2	.769	.363	.068	.284	.057	-	-
10	spec1.1	.376	.000	.068	.000	.308	-	-
4	truck1.1	.362	.083	.233	.047	.000	.000	.000
2	tank2.1	.356	.000	.066	.044	.000	.246	-

Table 4: MPR-MRF evidence for a tank in a FLIR image.

Index	Target	Evidence	1	2	3	4
1	tank1.1	.890	.338	.086	.103	.363
3	tank2.2	.781	.281	.089	.107	.304

Table 5: MPR-MRF evidence for an occluded "truck2".

Index	Target	Evidence	1	2	3	4	5	6	7
7	truck2.1	.668	.536	.0	.000	.132	-	-	-
8	truck2.2	.612	.000	.0	.184	.427	.0	.000	.0
4	truck1.1	.218	.000	.0	.000	.000	.0	.218	-

modest sized databases and correspondingly small search costs. The approach is also efficient because of the small sizes of the sector MRFs and their configuration spaces. The average processing time for our implementation of the system on a SPARCstation IPX was on the order of 20 seconds.

Though the results obtained so far are quite encouraging, the system is still a prototype and many improvements could be made in it, as regards both further DRC integration and more efficient implementation. The ability to obtain real FLIR data representing various controlled situations would greatly enhance developmental capabilities.

The EF could easily be modified to incorporate physical information about the expected objects and background, e.g. textural information about the background, or the expected contrast between the objects and the background (for FLIR, this depends on the time of day). The latter modification would provide a further basis for differentiating between real targets and decoys.

The system could be extended to handle compound objects in which the parts have different gray level ranges; for example, an MPR-MRF could first delineate the brightest parts, and HL information could then direct the establishment of additional centers to delineate other parts. Still another extension would involve modeling (simple or compound) ribbon-like objects, using pieces of medial axis instead of centers, and "radii" perpendicular to these axes.

The system could be generalized to 2.5D for range data and to 3D for CT images. Another possible extension involves using time-varying models to perform DRC on sequences of images.

References

- [1] A. Blake and A. Zisserman, *Visual Reconstruction*, MIT Press, Cambridge MA, 1987. Chapter 7: The Graduated Non-Convexity Algorithm.
- [2] L. D. Cohen, "On Active Contour Models and Balloons", *Computer Vision, Graphics, and Image Processing*, 53:211-218, 1991.
- [3] N. S. Friedland, "Utilizing Energy Function and Description Length Minimization for Integrated Delineation, Representation and Classification of Objects", University of Maryland Ph.D. Thesis, January 1993; Center for Automation Research Technical Report CAR-TR-657, University of Maryland, College Park, MD, 1993.
- [4] N. Friedland and D. Adam, "Automatic Ventricular Cavity Boundary Detection from Sequential Ultrasound Images Using Simulated Annealing", *IEEE Transactions on Medical Imaging*, 8:344-353, 1989.
- [5] N. S. Friedland and A. Rosenfeld, "Compact Object Recognition using Energy Function Based Optimization", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:770-777, 1992.
- [6] P. Fua and A. J. Hansen, "Objective Functions for Feature Discrimination: Theory", *Proceedings, DARPA Image Understanding Workshop*, pp. 443-460, 1989.
- [7] P. Fua and A. J. Hansen, "Objective Functions for Feature Discrimination: Applications to Semiautomated and Automated Feature Extraction", *Proceedings, DARPA Image Understanding Workshop*, pp. 676-694, 1989.
- [8] R. G. Gallager, *Information Theory and Reliable Communication*, p. 80, Wiley, New York, 1972.
- [9] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679-698, 1984.
- [10] M. Kass, W. Witkin and D. Terzopoulos, "Snakes: Active Contour Models", *International Journal of Computer Vision*, 1:321-331, 1988.
- [11] Y. G. Leclerc, "Image and Boundary Segmentation via Minimal-Length Encoding on the Connection Machine", *Proceedings, DARPA Image Understanding Workshop*, pp. 1056-1069, 1989.
- [12] S. Menet, P. Saint-Marc, and G. Medioni, "B-snakes: Implementation and Application to Stereo", *Proceedings, DARPA Image Understanding Workshop*, pp. 720-726, 1990.

Recognition with Local and Semi-local Invariants*

Ehud Rivlin and Isaac Weiss

Computer Vision Laboratory, Center for Automation Research,
University of Maryland, College Park, MD 20742-3275

Abstract

Geometric invariants are shape descriptors that remain unchanged under geometric transformations such as projection, or change of viewpoint. A new method of obtaining local projective and affine invariants is developed and implemented for real images. Being local, these invariants are much less sensitive to occlusion than global invariants. The computation of invariants is based on a canonical method. This consists of defining a canonical coordinate system using intrinsic properties of the shape, independently of the given coordinate system. Since this canonical system is independent of the original one, it is invariant and all quantities defined in it are invariant. The method is applied without the use of a curve parameter by fitting an implicit polynomial to a general curve in a neighborhood of each curve point. Several configurations are treated: a general curve without any correspondence, and curves with known correspondences of one or two feature points or lines. Experimental results for real 2D objects in 3D space are presented.

1 Introduction

Geometric invariants are shape descriptors which remain invariant under geometrical transformations such as projection or viewpoint change. They are important in object recognition because they enable us to obtain a signature of an object which is independent of external factors such as the viewpoint. In this paper we treat projective (viewpoint) and affine invariants in various geometrical configurations.

The subject of invariants has recently increased in importance and recognition in the vision community. Projective invariants were a very active mathematical subject in the latter half of the 19th century. However, in

vision only one projective invariant, the cross ratio [Duda and Hart 1973], was used until recently.

Projective invariants of curves and surfaces were first introduced in vision by the second author [Weiss 1988]. In that paper we reviewed both algebraic and differential methods for obtaining invariants and pointed out their usefulness for object recognition.

One can distinguish between two kinds of invariants: global and local. Global invariants describe a shape as a whole so they require knowledge about the whole shape. Examples are the moment invariants often used in the Euclidean case. Global (algebraic) projective invariants were described in [Weiss 1988]. They have been applied successfully by Forsyth *et al.* [1990, 1991] to industrial objects. Like any global descriptors, these quantities are quite sensitive to occlusion. Local (differential) invariants are more immune to this problem. They have been treated in [Weiss 1988, 1991]. So-called "mixed" invariants were developed by Van Gool *et al.* [1990], Barrett *et al.* [1990] and Bruckstein *et al.* [1991]. In this paper we develop both local and "mixed" invariants using a new approach that is simpler and more robust to noise than previous methods.

Local invariants can be defined at each point of a shape, and can be used to obtain a "signature" of that shape. In the Euclidean case, for instance, it is common to plot the curvature against the arclength, both of which are local Euclidean invariants. Such plots or "signatures" of curves can then be matched even if part of a curve is missing due to occlusion. We obtain such signatures in the projective and affine cases.

One can build an object recognition system that uses invariant "signatures" of curves, rather than the curves themselves, for matching. Therefore the matching does not require a search for the correct point of view. This is possible because of a general *completeness* property.

The completeness property of differential invariants can be described as follows. Given a plane curve and a transformation group, there are two independent invariants of the transformations at each point of the curve. These invariant functions contain all the information about the curve, except for the transformation to which they are invariant. Accordingly, given two invariants for each curve point, we can reconstruct the original curve up to a transformation belonging to the group.

More accurately, the following theorem holds [Guggen-

*The authors were supported in part by the Defense Advanced Research Projects Agency (ARPA Order No. 8459) and the U.S. Army Topographic Engineering Center under Contract DACA76-92-C-0009. The second author was also supported in part by the Air Force Office of Scientific Research under Grant F49620-92-J-0332.

heimer 1963, p. 144]: *All differential invariants of a (transitive) transformation in the plane are functions of two invariants of the lowest order and their derivatives.* Thus, given a curve, one can find a corresponding invariant curve, which we call its signature, that describes it uniquely, except for the relevant transformation.

This method applies to all kinds of local invariants. Projective and affine invariant signatures are used in [Weiss 1992] (with an explicit method) and [Weiss 1992b]. At each point of the given curve we calculate two invariants, I_1, I_2 . We plot these numbers as a point in an "invariant plane" whose coordinates represent invariants. In effect, we plot one invariant against the other. In this way the given curve maps into an invariant signature curve in the invariant plane. The signature uniquely identifies the curve regardless of the point of view.

Global invariants are often associated with algebraic methods and require no differentiation (although integration may be used for finding moments). Local invariants involve some form of differentiation. Larger transformation groups need higher orders of differentiation; projective invariants need a higher order of differentiation than affine, which in turn need a higher order than Euclidean.

We deal here mainly with curves. General curves can be treated in several ways. Two main camps exist among geometers: those who favor an explicit representation and those who prefer an implicit one. In the explicit method a curve is represented as functions of some parameter along the curve, e.g. $x(t), y(t)$. In the implicit approach a curve is represented by a relation $f(x, y) = 0$, without a parameter. The advantage of the implicit approach is that it does not require introduction of a parameter, which is not in fact part of the geometry of the curve itself. The relation between x and y is sufficient to completely characterize the curve. The explicit method makes it easier to obtain closed form formulas for general curves.

In finding invariants, the parameter is undesirable for the following reasons. The essence of finding invariants is the elimination of unknowns from the system, such as the unknown quantities describing the point of view. The parameter is also in general unknown since it can be chosen in an arbitrary way. It has to be eliminated so that the invariants will not depend on it. The more unknowns we have to eliminate, the more information we have to extract from the image, which translates in the explicit method to higher, and less reliable, derivatives.

Another reason to avoid the parameter is the quality of the fitting. In fitting a curve to data points, we make the assumption that the average squared distance is minimal, and the problem arises of how to estimate the distance of a point from a general curve. In the explicit method, the minimized functions are $x(t), y(t)$, measuring distances parallel to the x, y axes. These distances are very unstable when the curves themselves are almost parallel to the axes, and can introduce substantial errors. We also have to obtain two fitted functions $x(t), y(t)$ rather than one. An implicit fit minimizes the distance (roughly) *perpendicular* to the curves, and it thus seems more natural. It eliminates the parameter

before it enters the invariant expressions and adds to an accumulation of errors. In addition, the explicit method assumes the existence of some ordering among the data points so that a parameter can be assigned to them, which is not always possible.

These considerations are especially important in the projective case in which there is no natural parameter such as arclength. But even in the Euclidean and affine cases, which do admit a natural arclength, it needs to be obtained from the image and the same robustness considerations apply. The implicit method avoids the parameter altogether and thus increases robustness.

Most previous work on local invariants [Wilczynski 1906] was done using the explicit approach. An implicit approach was used by Halphen [1880] but it did not provide all the invariants and was cumbersome to implement. We present here a simple way of deriving local invariants in the implicit approach, without a curve parameter. The approach is based on transforming the shape to a canonical (intrinsic) system of coordinates, rather than obtaining closed form formulas for the invariants.

The canonical approach has another advantage for our purposes. A problem that arises in finding invariants is fitting a curve to the data points in an invariant way, i.e. the fitting method has to be invariant before the curve invariants can make sense. This is particularly true if the fitting error arises mainly not from random noise but from the shape itself, for instance when trying to fit a conic to a polygon. In previous methods [Forsyth 1991] invariant fitting could be done only in the affine case. The canonical method presented here is capable of obtaining an invariant fit in the general case.

Several kinds of situations will be treated here. The first involves general plane curves without any correspondence information. These require the highest number of derivatives so their signatures are the hardest to obtain. Next, shapes consisting of a curve and one known feature point will be treated. For the feature (or reference) point it is assumed that a correspondence can be established between two images. This enables us to eliminate some of the transformation parameters and reduce the amount of information needed from the curve itself, i.e. the orders of the derivatives. Using a curve and two reference points reduces this amount even further. The authors mentioned earlier treated these situations with the explicit approach, using derivatives with respect to a curve parameter. We will treat them here without a parameter. We will also treat curves with reference lines, which have not been previously treated, to our knowledge.

2 Finding Local Invariants—A Canonical Approach

In principle, one can find invariants of a curve f by one of the known methods. However, these invariants will not be local; they will depend on the window size and the curve order. In addition, the common methods are very cumbersome for high order curves and require the use of a symbolic manipulation program. Their robustness is

questionable.

Here we obtain these local invariants in a quite simple and intuitive way. The basic idea is to transform our coordinate system to a canonical one, i.e. a standard system which is defined by the intrinsic characteristics of the shape itself. Since this system is intrinsic, all quantities measured in it are independent of the initial system and are therefore invariants. One can give a simple example as follows: Given an image of a rod, we can calculate its length, which is a Euclidean invariant, by applying the formula for Euclidean distance. An alternative approach is to transform the coordinate system into a canonical one, in which the rod lies along the x axis and the origin is at one end of the rod. Then the x coordinate of the other end is the rod's length. We see that by moving to a canonical system we have obtained the invariant length without an explicit formula. The canonical system was determined by the properties of the shape rather than by some external factors.

An important differential example is finding Euclidean invariants of curves. We can move the coordinate system so that the x axis is tangent to the curve at some point that we choose on it, i.e. $y' = 0$ there. The second derivative y'' at this point is now equal to the curvature and is invariant since we obtain the same canonical system regardless of which system we started with. We see that by determining some of the properties of the system, the others are also determined and become invariant.

We generalize this approach to larger transformation groups. In general, the factors in a transformation can be eliminated by using the same kind of transformation, with the same number of factors, to go over to the canonical coordinates. The Euclidean invariants can be obtained by using a Euclidean transformation to obtain a Euclidean canonical system, etc.

The general projective transformation can be decomposed into simpler transformations: translation, rotation, skewing, scaling (making up the affine group), tilt and slant. We will use these to canonize the coordinates step by step. At each step some of the viewpoint parameters will be eliminated until we are left with a coordinate system independent of the original viewpoint and defined by the shape itself.

There are two basic requirements that the canonization process has to meet: it has to be *invariant*, i.e. produce a result that is independent of the original system, and it has to be *local*, i.e. independent of the exact fitting details such as the window size or fitted curve.

The Euclidean example above meets these requirements. The requirement of tangency is an invariant one, because the tangency property is unchanged under a projective transformation. The locality requirement is also met, because the tangency means that the first derivative dy/dx vanishes. A derivative is a local property and is independent of the size of the window in which it was calculated. It is also independent of exactly what curve was fitted (as long as the fit is good), because any fitted function can be expanded in a Taylor series with the same first derivatives.

For the Euclidean case we used the tangent to obtain a canonization process that met our requirements of in-

variance and locality. We can generalize the method by using an osculating curve, which is a generalization of the tangent. A tangent is a line having at least two points in common with the curve in an infinitesimal neighborhood, i.e. two "points of contact". This can be expressed as a condition on the first derivative. Similarly, a higher order osculating curve has more (independent) contact points, and the condition on the derivatives can be written as

$$\frac{d^k}{dt^k}(f^*(x, y) - f(x, y)) = 0, \quad k = 0 \dots n \quad (1)$$

with f^* being the osculating curve, f the given curve, and n the order of contact. Since the derivatives vanish, this condition is invariant to the parameter t . (We will derive the osculating curve without this parameter.) Since it has a geometric interpretation with points of contact, the condition is also projectively invariant. And since it is expressed as derivatives, it is also local. Thus all the independence requirements set forth earlier are met. (The derivatives will be calculated analytically from f .)

In the following sections we will use an osculating implicit curve f^* satisfying the above condition. This curve will be chosen as the simplest one that meets our needs; its shape is thus known. Thus it will be easier to handle than the original f which can be any function that fits. According to our needs we find either a cubic or a conic which osculates our fitted curve. We then transform the coordinates so that this cubic or conic takes on a particularly simple, predetermined form, i.e. we eliminate all its coefficients. In this new (canonical) system all quantities are invariants and we pick the ones that best suit our needs.

We will describe the correspondenceless case in full and summarize the other cases.

3 Local Projective Invariants Without Correspondence

We use the osculating curve method to eliminate all the projective unknowns and obtain two local invariants at any curve point. The outline of our method is as follows:

- Repeat the following steps for each pixel that belongs to the curve to obtain two independent invariants at that point of the curve:
 - Define a window around the pixel and fit an implicit polynomial curve to it, say a cubic or a quartic. All the following stages are performed analytically.
 - Derive a canonical, intrinsic coordinate system based invariantly on the properties of the shape itself, independently of the given coordinate system. By doing so we eliminate all the unknown quantities of the original system (the viewpoint). To accomplish this: define an "auxiliary curve" which osculates the original fitted curve with a known order of contact. The canonical system is defined so that in it the osculating curve has a particularly simple, predetermined form.

- Transform the original fitted curve to this new system. Since the system is canonical, all shape descriptors defined in it are independent of the original coordinate system and are therefore invariants. Pick two invariants that are independent of the window size or the order of the fitted curve, and depend only on the shape itself.

- Plot one invariant against the other to obtain an invariant signature curve. This is based on the completeness property discussed above.
- If an invariant fit is needed, we repeat the previous steps, i.e. redo the curve fitting in the new canonical system, and iterate until convergence.

In the following sections we describe the above steps in more detail.

3.1 Curve fitting

The method described above involves fitting an implicit curve to the available data points. To do so we have to determine parameters such as the order of the curve and the window size.

To determine the curve order, we need to know the minimum number of coefficients needed, or the amount of information that needs to be obtained from the image. To find invariants, we have to eliminate the information in the image which is specific to the coordinate system. For example, given a pencil that can move or rotate on a table, the position of the pencil and its orientation are not invariant but its length is a Euclidean invariant. Given the coordinates, say of the ends of the pencil, we can eliminate the position and orientation and calculate the length. Thus from the four measured coordinates we have eliminated the three Euclidean transformation parameters and found one invariant.

Similar arguments apply to other transformations. In the projective case, we want to eliminate eight parameters of the transformation, so the number of coefficients to be obtained from the image should exceed eight. Since we need two independent invariants at each pixel, we need ten independent quantities. A cubic has nine coefficients, but we also have the position of the point on the cubic for a total of ten quantities. Thus it is sufficient from purely geometrical considerations to fit a cubic to our data. However, other considerations push us towards a higher order curve.

We can see here an advantage over the explicit method that requires differentiation of $x(t)$, $y(t)$ with respect to the curve parameter t . The elimination argument above applies to this unknown parameter, i.e. this parameter has to be eliminated along with the coordinates, so that the invariants will be independent of it. This increases the amount of data that needs to be extracted from the image, e.g. the orders of the derivatives. In Wilczynski's method, the eighth derivatives of both x and y were needed, a total of 18 quantities. This reduced the reliability of the invariants. Thus avoiding the parameter from the outset reduces the number of quantities we need to obtain from the image and improves reliability.

Regarding the window size, we have found [Weiss 1991] that the wider the window, the more reliable the fitting

becomes. This is especially important in our case because of the relatively large number of independent quantities that we need to obtain, at least in the hardest case (the projective correspondenceless case). To maintain the accuracy of the fit in a wide window, a higher order curve has to be fitted. This prompts us to use quartic or higher curves, even though a cubic has enough parameters. The increased number of parameters needed for the higher order curves is not a problem because they do not all need to be independent; at most ten independent ones are needed.

In practice we have found it convenient to restrict ourselves to fourth order (quartic) curves although higher orders may be worth investigating. In the sequel we will deal with the fitted quartic

$$\begin{aligned} f(x, y) = & a_0 + a_1x + a_2y + a_3x^2 + a_4xy + a_5y^2 \quad (2) \\ & + a_6x^3 + a_7x^2y + a_8xy^2 + a_9y^3 + a_{10}x^4 \\ & + a_{11}x^3y + a_{12}x^2y^2 + a_{13}xy^3 + a_{14}y^4 = 0 \end{aligned}$$

with the cubic being the special case in which coefficients a_{10}, \dots, a_{14} vanish.

Once the curve order and window size have been chosen, the fitting itself can be done by standard methods. Simple least square fitting is quite ill conditioned because of the relatively large number of unknowns. The SVD (Singular Value Decomposition) method [Press *et al.* 1986] is very successful in overcoming this problem and we obtain a quite reliable fit.

We have thus obtained a local algebraic (parameterless) representation for the data. We will now find its invariants (analytically).

3.2 Deriving a canonical, intrinsic coordinate system

3.2.1 Euclidean canonization

First we detail the Euclidean canonization stage. As a convention, we denote the new coordinates after each canonization step by \bar{x}, \bar{y} and drop the bars before going to the next step, and similarly for other quantities.

The first step is translation, moving the origin to our curve point. Our pixel x_0, y_0 does not necessarily lie on the fitted curve but it is close to it. Thus, we find a point x_0, y_0^* which does lie on the curve, i.e. we solve eq. (3) for y_0^* , given x_0 . This is easy to do with Newton's method because y_0 is a close initial guess. We now translate the origin to x_0, y_0^* . (We could simplify the solution by first translating so that $x_0 = 0$ and then solving for y_0^* .) We drop the star from y^* . We now transform the curve coefficients to the new system and obtain new \bar{a}_i . This is done by expressing the old coordinates in terms of the new, $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{x}_0$, substituting in eq. (3) and rearranging. In this new system we have $\bar{a}_0 = 0$ which can be seen by simply substituting (0,0) in eq. (1).

The next step is to rotate the coordinates so that the x axis is tangent to the curve. It is easy to see that in the rotated system we must have $\bar{a}_1 = 0$ (because $df(x, y)/dx = 0$). To satisfy this condition we again express the old coordinates in terms of the new, with the rotation factor u_r :

$$x = \bar{x} + u_r \bar{y} \quad y = \bar{y} - u_r \bar{x} \quad (3)$$

Now a_1 is transformed to

$$\bar{a}_1 = a_1 - u_r a_2$$

To make this vanish we have to rotate by the amount

$$u_r = a_1/a_2$$

Since translation and rotation make up the Euclidean transformations, we have reached a Euclidean canonical system. All quantities defined in it are Euclidean invariants. The curvature at x_0 is now simply the second derivative, d^2y/dx^2 . The arclength is $|dx|$ since $dy = 0$.

3.2.2 Eliminating the projective unknowns

Of the eight parameters of the general projectivity we have already eliminated three by translation and rotation, so our osculating curve should have five coefficients, while passing through the origin and being tangent to the x axis. Following [Halphen 1880] we choose the "nodal cubic"

$$f^* = c_0 x^3 + c_1 y^3 + c_2 x y^2 + c_3 x^2 y + c_4 y^2 + x y = 0 \quad (4)$$

This curve intersects itself at the origin so it has two tangents there, one lying along the x axis. The other tangent is called the "projective normal" [Lane 1942]. Our treatment of the nodal cubic differs from Halphen's and yields the full range of invariants. (We also had the advantage of a symbolic manipulation program.)

Our goal is now to transform the coordinates so that this nodal cubic takes on the simple coefficient-free form

$$x^3 + y^3 + x y = 0 \quad (5)$$

It is known [Bronshtein 1985] as the *folium (leaf) of Descartes* (Figure 1). In a nutshell, we obtain it as fol-

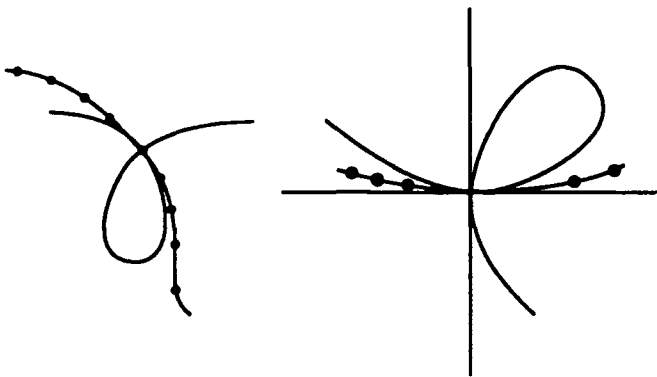


Figure 1: Osculating nodal cubic (left), folium of Descartes (right).

lows. We skew the coordinates so that the projective normal becomes perpendicular to the x axis, thus providing a canonical y axis. This eliminates c_4 . We scale the axes to eliminate c_0, c_1 , obtaining an affine canonical system with new \bar{c}_2, \bar{c}_3 . These are now *affine invariants*. We tilt and slant to eliminate them too, obtaining the projective canonical system.

We now find the nodal cubic f^* using the osculation condition, i.e. the equality of the first n derivatives of

f and f^* , eq. (1). The first derivative (and the zeroth) vanish because of the tangency to the x axis. To determine the five coefficients c_i we need five more derivatives to be equal, i.e. up to the sixth one. The condition of equal derivatives ensures the locality of the treatment and also its invariance, as discussed earlier.

To go further, we need to calculate the derivatives $d^n y/dx^n$ of the fitted curve. This is done analytically from $f(x, y)$. To do it we use the fact that all the derivatives of f vanish, since f vanishes identically (eq. 1). The first derivative, for example, is

$$\frac{df}{dx} = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} \frac{dy}{dx} = 0$$

This is a linear equation for dy/dx . It is superfluous because we have already demanded its vanishing (tangency). However, each successive differentiation gives one linear equation for one higher $y^{(n)}$ in terms of lower derivatives. The calculation is tedious and we used a symbolic manipulation program to calculate up to $y^{(8)}$ in terms of the a_i .

Setting $a_2 = 1$ and denoting $d_n = \frac{1}{n!} \frac{d^n y}{dx^n}(0)$ we have

$$d_2 = -a_3 \quad (6)$$

$$d_3 = -a_6 - d_2 a_4 \quad (7)$$

$$d_4 = -a_{10} - d_2 a_7 - d_2^2 a_5 - d_3 a_4 \quad (8)$$

$$d_5 = -d_2 a_{11} - d_2^2 a_8 - d_3 a_7 - 2d_2 d_3 a_5 - a_4 d_4 \quad (9)$$

$$d_6 = -d_2^2 a_{12} - d_3 a_{11} - d_3^2 a_9 - 2d_2 d_3 a_8 - d_4 a_7 - a_4 d_5 + (-2d_2 d_4 - d_3^2) a_5 \quad (10)$$

Given these derivatives we find the coefficients c_n of the nodal cubic (4) as follows. We write the nodal cubic as

$$y(x) = \sum_{n=0}^6 d_n x^n$$

and substitute it in the cubic expression, eq. (3). Collecting terms with the same power x^n we obtain five equations for the five c_i in terms of the d_n . Their solution is

$$c_0 = -d_2 \quad (11)$$

$$c_1 = \frac{d_4^2(d_4 d_6 - d_2^2) + d_2^3(-d_2^2 d_6 + 2d_3 d_4 d_5 - 2d_4^3) + 3d_2^2 d_3^2 d_4^2 - 3d_2 d_4^2 d_6 + d_4^3}{d_2^2 d_6 - 3d_2^2 d_3 d_4 + 2d_2^2 d_3^2} \quad (12)$$

$$c_2 = \frac{d_2^3(d_3 d_6 - d_4 d_5) + d_2^2(d_3 d_4^2 - 3d_3^2 d_5) + 5d_2 d_3^2 d_4^2 - 3d_3^3}{d_2^2 d_6 - 3d_2^2 d_3 d_4 + 2d_2^2 d_3^2} \quad (13)$$

$$c_3 = \frac{d_2^3 d_6 + d_2^2(-5d_3 d_5 - 2d_4^2) + 13d_2 d_3^2 d_4^2 - 7d_4^3}{d_2^2 d_6 - 3d_2^2 d_3 d_4 + 2d_2^2 d_3^2} \quad (14)$$

$$c_4 = -\frac{d_2^3 d_6 + d_2^2(-4d_3 d_5 - 2d_4^2) + 10d_2 d_3^2 d_4^2 - 5d_4^3}{d_2^2 d_6 - 3d_2^2 d_3 d_4 + 2d_2^2 d_3^2} \quad (15)$$

Having found the coefficients c_i , we proceed to eliminate them. First, we orthogonalize the axes, i.e. skew the system so that the two nodal tangents become perpendicular. This will eliminate the c_4 term in the nodal cubic (4). Our skewing transformation is

$$x = \bar{x} + u_r y$$

with $u_s = -c_4$ being the skewing factor. y remains unchanged. Substituting the above equation in the cubic (4) and rearranging we obtain new coefficients

$$\bar{c}_1 = -c_0 c_4^3 + c_3 c_4^2 - c_2 c_4 + c_1 \quad (16)$$

$$\bar{c}_2 = 3c_0 c_4^2 - 2c_3 c_4 + c_2 \quad (17)$$

$$\bar{c}_3 = c_3 - 3c_0 c_4 \quad (18)$$

We again drop the bars from c_i and x .

One advantage of the orthogonalization is that it makes it possible to decouple the next transformations, i.e. the slantings and scalings in the x and y directions. We can now proceed with these transformations in any order to eliminate the remaining c_i .

We next scale the axes with the scaling factors s_x, s_y :

$$x = \bar{x}/s_x, \quad y = \bar{y}/s_y \quad (19)$$

where $s_x = c_0^{2/3} c_1^{1/3}$, $s_y = c_0^{1/3} c_1^{2/3}$. Substituting this in the orthogonalized cubic we obtain

$$\bar{x}^3 + \bar{y}^3 + \bar{c}_2 \bar{x} \bar{y}^2 + \bar{c}_3 \bar{x}^2 \bar{y} + \bar{x} \bar{y} = 0 \quad (20)$$

with

$$\bar{c}_2 = \frac{c_2}{s_y}, \quad \bar{c}_3 = \frac{c_3}{s_x}$$

These quantities are *local affine invariants* because we have reached an affine canonical system. We have used all possible affine transformations (translation, rotation, skewing, scaling) to eliminate all possible affine transformation factors and arrive at the above form of the cubic, so the remaining coefficients are uniquely defined regardless of which system we started with.

A projective canonical system is obtained by eliminating the last two coefficients using slants, which are purely projective, in the x and y directions. To do this, we drop the bars from the last cubic form (20), and substitute x, y in terms of the projective canonical \bar{x}, \bar{y} :

$$x = \frac{\bar{x}}{1 + \sigma_x \bar{x} + \sigma_y \bar{y}} \quad y = \frac{\bar{y}}{1 + \sigma_x \bar{x} + \sigma_y \bar{y}} \quad (21)$$

with the x - and y -slant factors

$$\sigma_x = -c_3, \quad \sigma_y = -c_2$$

This finally brings us to Descartes' folium, eq. (5).

This concludes the elimination of the cubic coefficients and brings us to the projective canonical system. This system was defined invariantly by intrinsic properties of the curve such as the shape of the osculating nodal cubic, which is independent of the original coordinate system.

3.3 Projective invariants

We now have an invariant canonical system and affine invariants, but still no projective invariants. To obtain them, we transform the original fitted curve f , eq. (1), to our canonical system. We collect all the transformations that were performed during the canonization process. We have already translated and rotated f (with the factors x_0, y_0, u_r); we now perform the rest of the transformations making up the projectivity (with factors $u_s, \sigma_x, \sigma_y, s_x, s_y$) on f . The coefficients of f transform to new ones \bar{a}_i , which are now all invariants because they represent a fitted curve defined in the invariant system.

The only remaining question is how to select functions of the invariants \bar{a}_i which best suit our needs.

As mentioned before, the condition of locality dictates that we use derivatives of the curve rather than some arbitrary functions of the \bar{a}_i . The first six derivatives at x_0 are already determined by the canonization process (as $d_0, \dots, d_6 = 0, 0, -1, 0, 0, 1, 0$). Thus we need the seventh and eighth derivatives. These can be obtained in this particular system similarly to eqs. (6)–(10). With the above values of d_n we have (dropping the bars)

$$I_1 = d_7 = a_{13} - a_7 + 2a_5 \quad (22)$$

$$I_2 = d_8 = -a_{14} - a_{11} + 2a_8 - a_4 d_7 \quad (23)$$

These quantities are our *local projective invariants*.

In conclusion, we started with a curve fitted to data points around x_0, y_0 , and after a series of transformations of this curve we arrived at local invariants which are independent of the fitting details or the point of view. We can repeat the process for other points to obtain an invariant signature. No correspondence is needed.

3.4 Experimental implementation

The above method was implemented to extract local invariants from a set of real images. Each image was processed to obtain a contour curve for the relevant object, using standard techniques of edge detection and thinning. We used a window about 50 pixels wide around each contour point and fitted an implicit curve to it, minimizing the square distances with SVD. The coefficients of this fitted curve were used to calculate the invariants.

Figure 2 shows two views of a hanger. Effects of perspective distortion can be seen. Figure 3 shows the hanger under partial occlusion. Figures 4, 5, and 6 show the local invariants for the three hanger images. A good match of the signatures is obtained. A check for a match is demonstrated in Figure 7. The match is between the hangers in Figure 4 and Figure 6, where it is partially occluded. It can be seen that the occlusion does not prevent us from obtaining a good signature. We should mention that symmetry helps in getting a full signature for the hanger. For asymmetric objects only part of the signature is obtained.

Figure 8 shows a different object, a coat rack, from two different viewpoints. We used the parts of the rack on which coats hang. These parts are somewhat similar in character to the hanger (under projectivity). Accordingly, the signature has some similarity to the previous one but it is different enough to distinguish the hanger from the coat rack.

The invariant signatures are presented (one on top of the other) in Figure 9. The local invariants obtained from the coat rack (Figure 8) are compared with those of the first hanger image (Figure 2). The result of this comparison is presented in Figure 10.

4 Local Invariants With Some Correspondence

While the previous process does not require correspondence, it leads to fitting rather high order curves, which may be sensitive to noise. This problem is discussed in

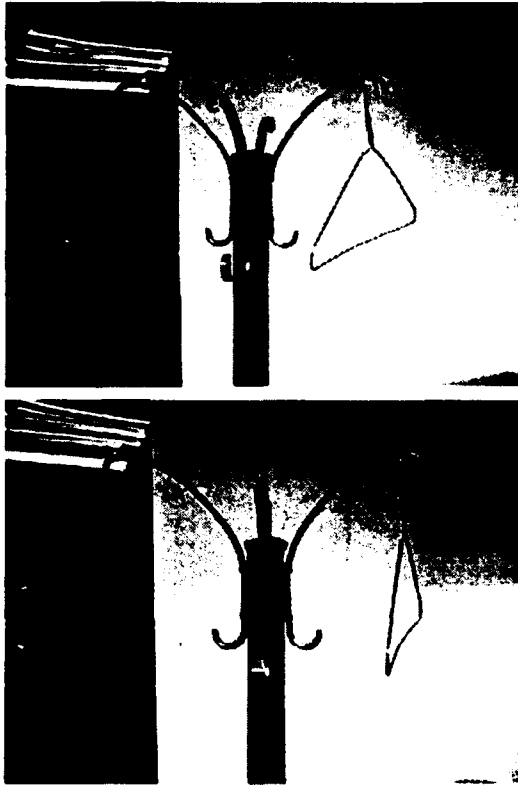


Figure 2: Two views of a hanger.

[Weiss 1991] and it is shown that one way of overcoming it is to use a wide window.

Another approach to increasing robustness is to use some reference features, e.g. points or lines, for which the correspondence is known. For example, a silhouette of an airplane can contain both curved parts and straight lines. We can use this information to eliminate some of the parameters of the projective or affine transformation; fewer curve descriptors will be needed for the elimination of the remaining ones. Invariants involving both derivatives and reference points were found by Barrett *et al.* [1990] and Van Gool *et al.* [1990]. However, they still use a curve parameter t which also has to be eliminated, and this reduces the robustness of their method.

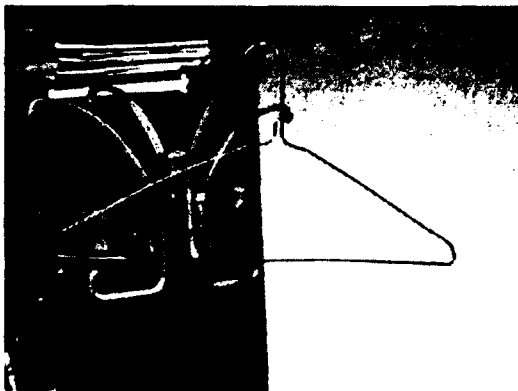


Figure 3: The hanger under partial occlusion.

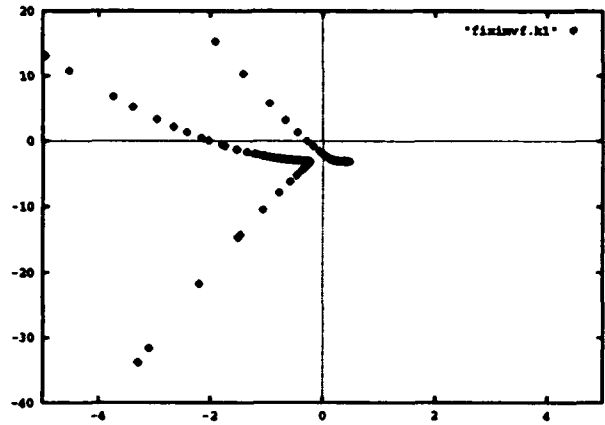


Figure 4: The invariant signature of the first hanger image.

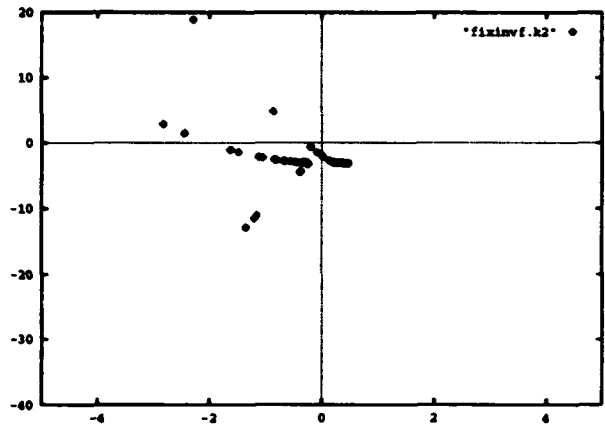


Figure 5: The invariant signature of the second hanger image.

The “parameterless” method described above is perfectly suited for this situation, and again leads to a reduction in the number of quantities needed from the image and to increased reliability. Here we use a canonical method similar to that used in the correspondenceless case in order to find local invariants while avoiding the curve parameter. This makes the method more robust as there are fewer unknowns to eliminate. In addition, as in the previous case, the canonical frame makes it possible to obtain an invariant fit using an iterative process, which should increase the robustness further.

The first stage is similar to the previous case: fit a high order curve over some window around some x_0, y_0 and then translate and rotate until the origin is at x_0, y_0 and the x axis is tangent to the curve. We need a smaller window than before and a lower order curve because we need only lower derivatives.

Again we obtain an auxiliary osculating curve that will help us find the canonical system. However, we do not need the nodal cubic; a conic, with three parameters, will suffice in all cases:

$$f^* = c(x, y) = c_0x^2 + c_1y^2 + c_2xy + y = 0 \quad (24)$$

The exact process of finding the conic and canonizing differs in each case. However, the principles of invariance and locality must be maintained. In the following

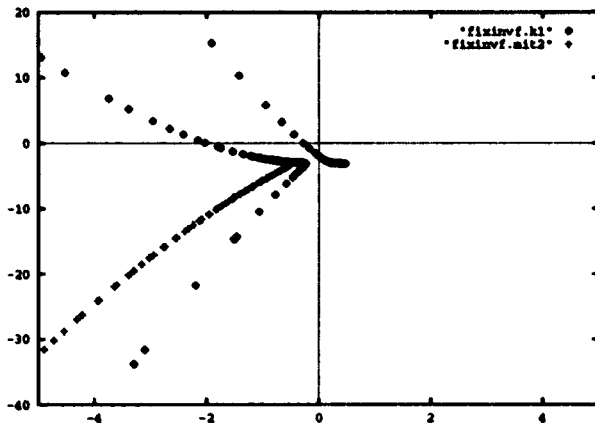


Figure 10: The invariant signature of the second object (the coat rack) presented on top of the signature of the first object (the hanger).

- **A Curve and Two Feature Points:**

This case requires only the second derivative to determine the osculating conic, rather than the fourth as before. We first find the conic that osculates the fitted curve with second order contact, and also passes through the two reference points. This uniquely determines the conic. We then find the line that passes through the two reference points. This brings us to the same situation as before, namely a conic plus a line, but with two fewer derivatives.

- **A Curve and Two Feature Lines:**

This case requires only the second derivative to determine the osculating conic, rather than the fourth as before. We first find the conic that osculates the fitted curve with second order contact, and is also tangent to the two reference lines. We then find the intersection point of the reference lines. This brings us to the case of a conic plus a point that we dealt with before, but with two fewer derivatives.

- **A Curve, a Point and a Line:**

As before we require that the conic osculate the fitted curve up to second order contact. In addition we require that the reference line be polar to the reference point w.r.t. the conic. This provides enough conditions to determine the conic. Achieving this brings us again to the situation of a conic plus a point, to be canonized as before, again with two fewer derivatives.

In what follows we describe the above processes in detail, and also give experimental results for some of the cases.

4.1 Transforming to a Euclidean canonical system

In all of the above processes the reference points and lines need to be transformed to the Euclidean canonical system. For a feature point x_1, y_1 the transformation is

$$\bar{x}_1 = (x_1 - x_0 - u_r(y_1 - y_0))/(1 + u_r^2) \quad (25)$$

$$\bar{y}_1 = (y_1 - y_0 + u_r(x_1 - x_0))/(1 + u_r^2) \quad (26)$$

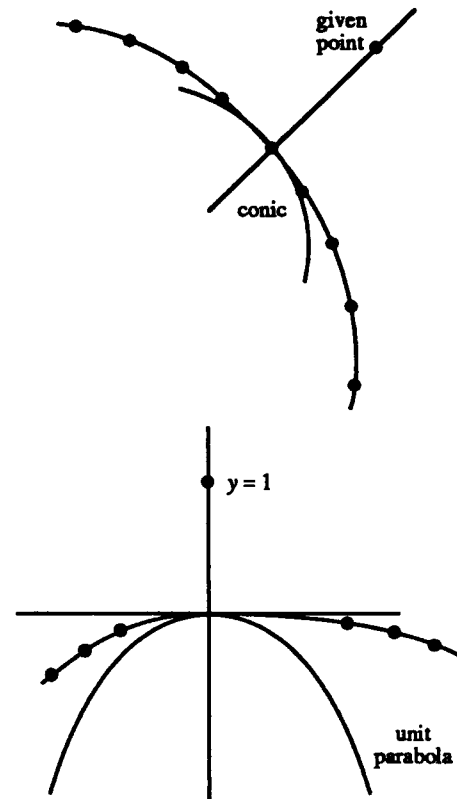


Figure 11: On the left, an osculating conic. On the right, the canonical conic and point.

(This involves the inverse of the rotation of the curve f , eq. (3), because points transform with the inverse of the curve transformation.)

The reference (feature) line $b_0 + b_1x + b_2y$ is translated and rotated as

$$\bar{b}_0 = b_0 + b_1x_0 + b_2y_0 \quad (27)$$

$$\bar{b}_1 = b_1 - u_r b_2 \quad (28)$$

$$\bar{b}_2 = b_2 + u_r b_1 \quad (29)$$

We again drop the bars from all quantities.

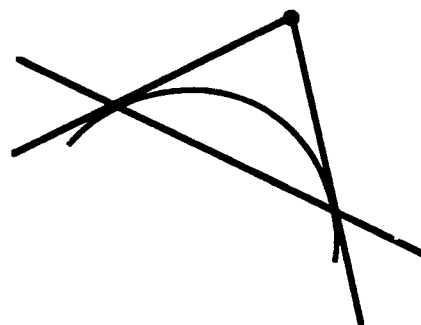


Figure 12: The relation between a polar line and a point.

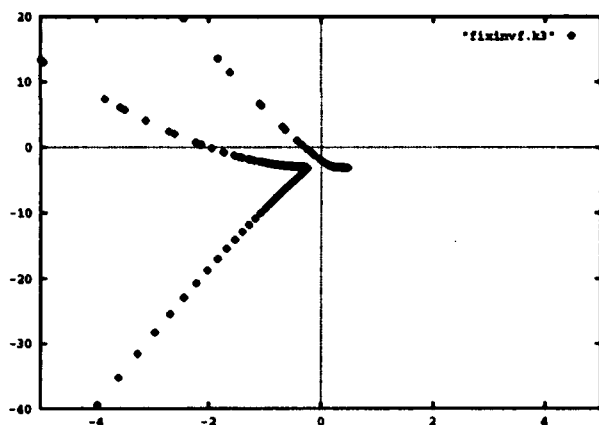


Figure 6: The invariant signature of the occluded hanger image.

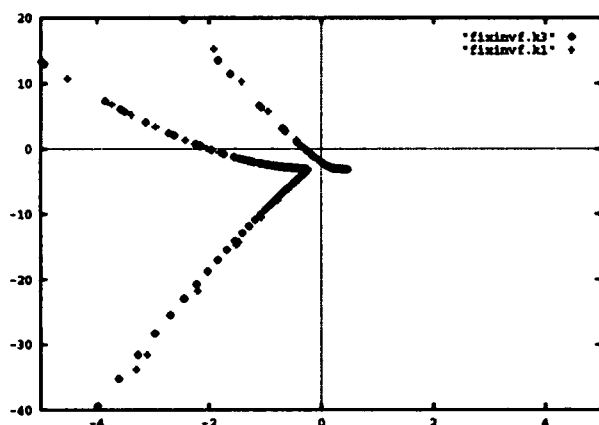


Figure 7: The invariant signature of the occluded hanger image presented on top of the signature of the unoccluded hanger.

we will briefly describe the processes for the different possible combinations. Each known feature point or line reduces the number of derivatives needed by two, because it eliminates two transformation factors.

- **A Curve and One Feature Point:**

We draw a line joining the given reference point x_1, y_1 to the curve point x_0, y_0 (Figure 11). This is obviously a projectively invariant operation. We use this line as our new y axis. As before we skew the system so this line becomes perpendicular to x . We thus obtain an orthogonal system which we can scale and slant as before.

To do this, we obtain an osculating conic to our fitted curve f . We need only fourth order contact, rather than sixth as before.

After fitting the conic, our goal will be to transform to a canonical system in which the conic is a unit parabola $x^2 + y = 0$, and the distance between the curve point and the reference point is unity (right hand side of Figure 11).

- **A Curve and One Feature Line:**

We convert to the previous case by finding the polar point of the given line with respect to the osculating

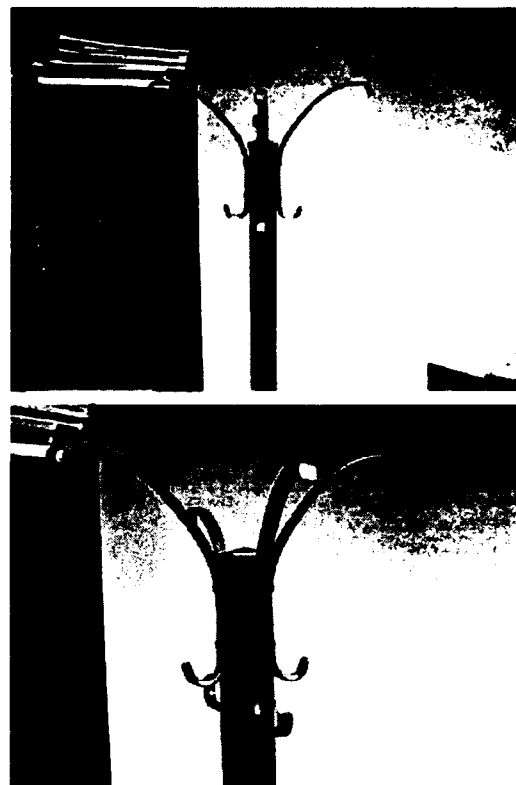


Figure 8: Two views of a coat rack.

conic. Polarity of a point and a line is an invariant relation. Given a point, we can draw from it two tangents to the conic, creating two points at which these tangents touch the conic. The line joining these two points is the polar line of the given point w.r.t. the conic (Figure 12).

The conic is found in the same way as in the previous case, requiring osculation in the fourth derivatives. After obtaining the polar point in a Euclidean canonical system, we are in the same situation as in the previous case, having a conic and a point, and we can proceed to find invariants as before.

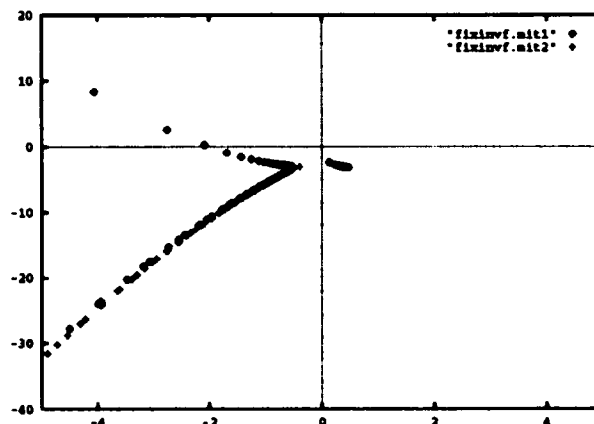


Figure 9: The invariant signatures of the coat rack. The signatures are presented one on top of the other.

4.2 A Curve and one feature point

We first find the first four derivatives of f using eqs. (6)–(10). From these we find the coefficients of the osculating conic by the same method we used for the nodal cubic. The result is

$$c_0 = -d_2 \quad (30)$$

$$c_1 = -(d_2 d_4 - d_3^2)/d_2^3 \quad (31)$$

$$c_2 = -d_3/d_2 \quad (32)$$

To orthogonalize the system, we want to obtain $\bar{x}_1 = 0$. This is achieved by skewing (eq. 19) with the skewing factor $u_s = x_1/y_1$. The orthogonalization changes the conic coefficients to

$$\bar{c}_1 = c_1 + c_0 u_s^2 + c_2 u_s \quad (33)$$

$$\bar{c}_2 = c_2 + 2c_0 u_s \quad (34)$$

We drop the bars from the c_i . The reference point coordinates are now $(0, y_1)$.

For the affine case we only need scaling, eq. (19).

It is easy to obtain a distance of unity between the origin and the reference point by scaling the y axis with $s_y = \pm 1/y_1$. (The sign is taken to be the same as the sign of c_0 .) Scaling in the x direction is done by requiring $c_0 = 1$, which is achieved by $s_x = \sqrt{c_0 s_y}$. Substituting the scaling transformation (19) in the conic (24) we obtain (dropping bars)

$$x^2 + \frac{c_1}{s_y} y^2 + \frac{c_2}{s_x} xy + y = 0$$

The two remaining coefficients, c_1/s_y and c_2/s_x , are affine invariants. (The conic here is not a unit parabola but has these two invariant coefficients.)

For projective invariants, we first have to slant the shape in the x and y directions. (This has to be done *before* scaling.) The terms containing c_1 and c_2 are eliminated using the transformation (21) with the x, y slant factors being $\sigma_x = -c_2$, $\sigma_y = -c_1$.

As in the affine case we use the reference point for scaling, but now its distance has changed because of the slant. The new distance is now $y^p = y_1/(1 - \sigma_y y_1)$. We want to scale y so that this distance is unity, so $s_y = \pm 1/y^p$ (again with the sign of c_0).

At this point the conic is reduced to $c_0 x^2 + y/s_y = 0$. To obtain a unit parabola and get rid of c_0 we scale in the x -direction with $s_x = \sqrt{c_0 s_y}$.

We have thus obtained the projective canonical system. To obtain the invariants, we have to transform the original fitted curve f to this system. Again all the transformed a_i are invariants, but we need the ones that are local in nature and independent of the fitting details, namely derivatives. Since we have used up the first four derivatives we need the fifth and sixth (two less than in the correspondenceless case). To obtain them we substitute in the expressions (6)–(10) the canonical values $d_0, \dots, d_4 = 0, 0, -1, 0, 0$ and obtain

$$I_1 = d_5 = a_{11} - a_8 \quad (35)$$

$$I_2 = d_6 = a_9 - a_{12} - a_4 d_5 \quad (36)$$

These are our local projective invariants.

4.3 A curve and one feature line

The conic is found in the same way as in the previous case, requiring osculation in the fourth derivatives. The polar line is found as follows:

Given a point x_1^h in homogeneous coordinates, we can write the coefficients b_i of its polar line with respect to a homogeneous conic

$$C = c_0(x^h)^2 + c_1(y^h)^2 + c_2 x^h y^h + y^h z^h = 0$$

as

$$b_0 = \frac{\partial C}{\partial x^h} \Big|_{x_1^h} = y_1^h \quad (37)$$

$$b_1 = \frac{\partial C}{\partial y^h} \Big|_{x_1^h} = 2c_0 x_1^h + c_2 y_1^h \quad (38)$$

$$b_2 = \frac{\partial C}{\partial z^h} \Big|_{x_1^h} = 2c_1 y_1^h + c_2 x_1^h + z_1^h \quad (39)$$

(C is first differentiated and then the point coordinates x_1^h are substituted in the right hand side.) In our case we know the line b_i and the conic C in the above equation, so we have a set of linear equations for the point x_1^h . Solving these equations we obtain

$$x_1^h = -b_1 + c_2 b_0 \quad (40)$$

$$y_1^h = -2c_0 b_0 \quad (41)$$

$$z_1^h = b_1 c_2 - 2c_0 b_2 + (4c_0 c_1 - c_2^2) b_0 \quad (42)$$

Going back to regular coordinates we have the polar point in our Euclidean canonical system

$$x_1 = x_1^h/z_1^h, \quad y_1 = y_1^h/z_1^h$$

We are now in the same situation as in the previous case, having a conic and a point in a Euclidean canonical system, and we can proceed to find invariants as before.

4.4 A curve and two feature points

We need here the formula for the conic coefficients in terms of the second derivative and the reference points:

$$c_0 = -d_2 \quad (43)$$

$$c_1 = \frac{c_0(x_1 x_2^2 y_1 - x_1^2 x_2 y_2) - y_1(x_2 y_2 - x_1 y_2)}{x_2 y_1^2 y_2 - x_1 y_1 y_2^2} \quad (44)$$

$$c_2 = \frac{-c_0(x_2^2 y_1^2 - x_1^2 y_2^2) + y_1 y_2^2 - y_1^2 y_2}{x_2 y_1^2 y_2 - x_1 y_1 y_2^2} \quad (45)$$

where x_1, y_1, x_2, y_2 are the reference point coordinates in the Euclidean canonical system.

Here (43) is the same condition on c_0 as in all previous cases, and (44)–(45) are obtained by substituting the reference points in the conic (24) and solving for c_1 and c_2 .

The affine invariants are calculated from the c_i as in the previous case. The projective invariants are now the third and fourth derivatives, two lower than before. Substituting $d_2 = -1$ in eq. (6) we obtain

$$d_3 = -a_6 + a_4 \quad (46)$$

$$d_4 = -a_{10} + a_7 - a_5 - a_4 d_3 \quad (47)$$

which are our local projective invariants.

4.5 A curve and two feature lines

The only new thing here is finding the conic. The tangents to a conic satisfy the equations of the "line conic", which is the dual of a regular conic. When representing the conic in matrix notation, the line conic matrix is the inverse of the point conic matrix. The inverse matrix of (24) is

$$C^{-1} = \begin{pmatrix} 1 & 0 & -c_2 \\ 0 & 0 & 2c_0 \\ -c_2 & 2c_0 & c_2^2 - 4c_0c_1 \end{pmatrix}$$

c_0 is determined as before by the second derivative d_2 . The reference lines satisfy the equations $bC^{-1}b^t = 0$, from which c_1 and c_2 can be found. We obtain the conic

$$c_0 = -d_2 \quad (48)$$

$$c_2 = \frac{(b_0'^2 b_1^2 - b_1'^2 b_0^2)/2 + 2c_0(b_2 b_0 b_0'^2 - b_2' b_0' b_0^2)}{b_1 b_0 b_0'^2 - b_1' b_0' b_0^2} \quad (49)$$

$$c_1 = \frac{b_1^2 - 2c_2 b_0 b_1 + 4c_0 b_0 b_2 + c_2^2 b_0^2}{4c_0 b_0^2} \quad (50)$$

where b_i, b_i' are the coefficients of the two reference lines in the Euclidean canonical system.

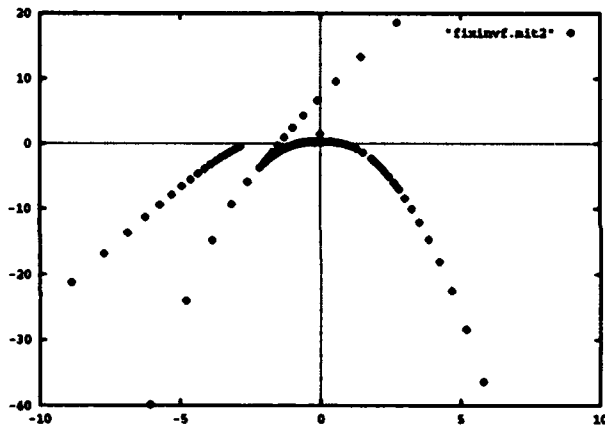


Figure 13: The invariant signature of the coat rack image.

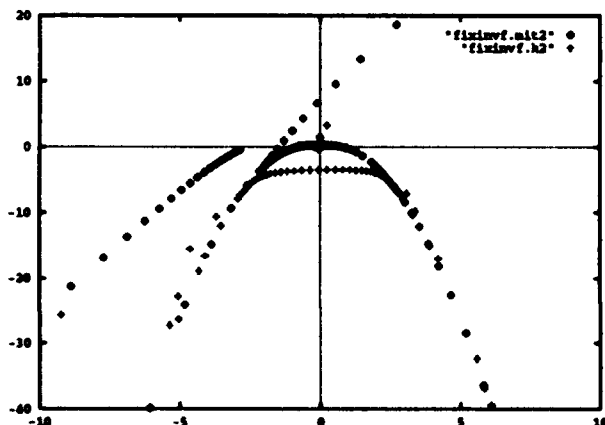


Figure 14: The invariant signatures of the images of the hanger and the coat rack presented on top of each other.

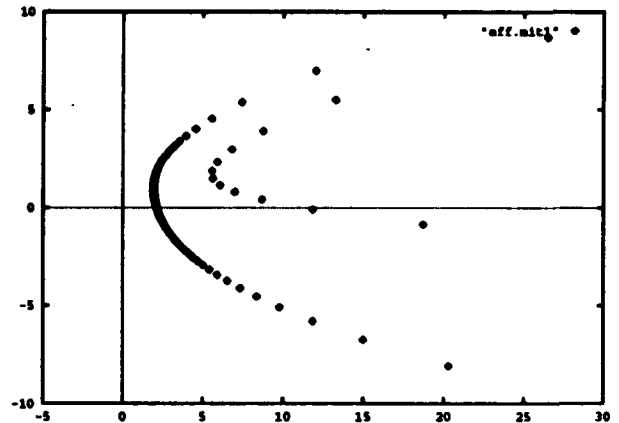
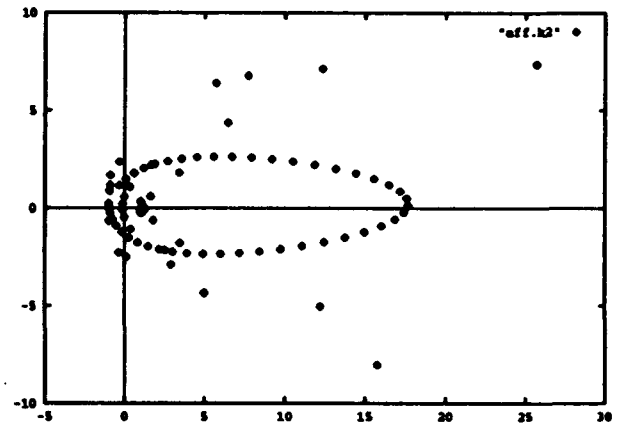


Figure 15: The affine invariant signatures of the hanger and the coat rack.

Experiments

The images of the hanger and the coat rack were used to derive local signatures using two feature lines. The signature obtained from the coat rack image is presented in Figure 13. A comparison of the two signatures for the hanger and the coat rack is presented in Figure 14.

The curve and two feature lines method was used to achieve affine invariants for the same objects. The results of the invariant computation are presented in Figure 15. A comparison of the invariants of the two objects is shown in Figure 16.

4.6 A curve, a point and a line

We require that the conic osculate the fitted curve up to second order contact. In addition we require that the reference line be polar to the reference point w.r.t. the conic. This provides enough conditions to determine the conic. Achieving this will bring us again to the situation of a conic plus a point, to be canonized as before, again with two fewer derivatives.

As before, the osculation condition leads to $c_0 = -d_2$. Setting $z_1^h = 1$, the first of the polar equations (37) leads to $y_1 = b_0$, and the line coefficients have to be normalized so that this equation is satisfied. This leads to the substitution

$$b_1 \leftarrow b_1 y_1 / b_0, \quad b_2 \leftarrow b_2 y_1 / b_0$$

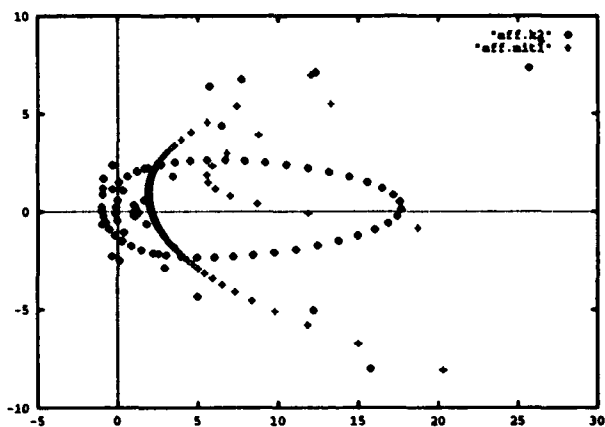


Figure 16: The affine invariant signatures of the hanger and coat rack images presented on top of each other.

The remaining two polar equations are then

$$2c_0x_1 + c_2y_1 = b_1 \quad 2c_1y_1 + c_2x_1 + 1 = b_2$$

which are satisfied by the conic coefficients

$$c_1 = ((b_2 - 1)y_1 + 2c_0x_1^2 - b_1x_1)/(2y_1^2) \quad (51)$$

$$c_2 = -(2c_0x_1 - b_1)/y_1 \quad (52)$$

The affine and projective invariants are calculated as in the previous two cases.

5 Conclusions

We have presented a method for finding local projective and affine invariants, and have applied it to real images. The method consists of defining a canonical coordinate system using intrinsic properties of the shape, independently of the given coordinate system. Since this canonical system is independent of the original one, it is invariant and all quantities defined in it are invariant. The method is general and can be used locally or globally, implicitly or explicitly. We have applied the method to find local invariants of a general curve without any correspondence, and of curves with known correspondences of one or two feature points or lines. Experimental results for both cases are presented.

Our method combines the advantages of the algebraic and differential methods. The differential method, being local, is much less sensitive to occlusion. The algebraic (implicit) method has an advantage in robustness because it does not need to eliminate an unknown curve parameter. Our experiments with real images have shown that, by using our local implicit method, we can find an invariant signature which is both insensitive to occlusion and relatively reliable. We have also demonstrated that these signatures, while unchanged under changes in viewpoint, do differ for images of even slightly different objects. Thus they have enough descriptive power to distinguish between many different kinds of objects. Therefore they can be used in an automated object recognition system that can distinguish and identify objects regardless of the point of view from which they are observed.

References

- [1] Barrett, E., Payton, P., Haag, N. and Brill, M. [1991], "General methods for determining projective invariants in imagery", *CVGIP:IU* 53, 45-65.
- [2] Bronshtein, I.N. and Semendyayev, K.A. [1985], *Handbook of Mathematics*, Van Nostrand, New York.
- [3] Bruckstein, A. and Netravali, A. [1990], "On differential invariants of planar curves and recognizing partially occluded planar objects", AT&T TR, July 1990.
- [4] Proc. DARPA-ESPRIT Workshop on Invariance [1991], Reykjavik, Iceland, March 1991.
- [5] Duda, R.O. and Hart, P.E. [1973], *Pattern Recognition and Scene Analysis*, Wiley, New York.
- [6] Forsyth, D., Mundy, J.L., Zisserman, A., Coelho, C., Heller, A., and Rothwell, C. [1991], "Invariant descriptors for 3-D object recognition and pose", *IEEE-PAMI* 13, 971-991.
- [7] Guggenheimer, H. [1963], *Differential Geometry*, Dover, New York.
- [8] Halphen, M. [1880], "Sur les invariants différentiels des courbes gauches", *J. Ec. Poly.*, XXVIII, 1.
- [9] Meer, P. and Weiss, I. [1992], "Smoothed differentiation filters for images", *J. Visual Communication and Image Representation* 3, 58-72.
- [10] Press, W.H., Flannery, B.P., Teukolsky, S.A., and Wetterling, W.T. [1986], *Numerical Recipes*, Cambridge University Press, Cambridge, UK.
- [11] Van Gool, L., Wagemans, J., Vandeneede, J., and Oosterlinck, A. [1990], "Similarity extraction and modeling", *Proc. 3rd International Conference on Computer Vision*, 530-534.
- [12] Weiss, I. [1988], "Projective invariants of shapes", *Proc. DARPA Image Understanding Workshop*, 1125-1134.
- [13] Weiss, I. [1991], "High order differentiation filters that work", University of Maryland, Center for Automation Research, CAR-TR-545, March 1991.
- [14] Weiss, I. [1992], "Noise resistant projective and affine invariants", *Proc. DARPA Image Understanding Workshop*, 683-692.
- [15] Weiss, I. [1992b], "Local projective and affine invariants", University of Maryland, Center for Automation Research, CAR-TR-612, April 1992.
- [16] Wilczynski, E.J. [1906], *Projective Differential Geometry of Curves and Ruled Surfaces*, Teubner, Leipzig.

Constraint Processing Applied to Industrial Inspection and Continuous Product Improvement

J. Alison Noble and Joseph L. Mundy
GE Corporate Research and Development Center
P.O. Box 8, 1 River Road
Schenectady, NY, 12301, USA.
tel: (518)-387-6723 fax: (518)-387-5752
email: nobleja@crd.ge.com

Abstract

This paper describes an object-oriented machine vision software system for industrial part inspection and manufacturing process parameter understanding and monitoring being developed at the GE Corporate Research & Development Center. Key concepts in this system have been adapted from defense-related image understanding research, including geometric constraint programming and object-oriented design in machine vision. We describe the implementation of a working system for part inspection from X-ray images. We outline some new results in the area of dimension (parameter) estimation in the presence of measurement uncertainty using Bayesian techniques to compute part dimension distributions from image sequences. We indicate some new directions for our work which address central issues in application-driven computer vision.

Keywords: *object-oriented methodology, constraint processing, deformable templates, automated 2D image analysis, dual-use technology.*

1 Introduction

This paper describes an on-going effort at GE Corporate Research & Development to transfer recent advances in model-based vision and object recognition to industrial machine vision applications. Industrial part design and manufacture offers a wealth of opportunities to address many of the central issues in computer vision research including defect classification problems (feature enhancement and signal processing), part tolerancing and measurement error analysis (modelling using constraints with uncertainty), object-oriented design for machine vision software (IU standards), multi-modality image analysis and fusion, and performance assessment of image analysis algorithms (inspection/recognition system accuracy). We have focussed on two of these themes, geometric constraint-processing and object-oriented machine vision software, and are applying them to build an X-ray inspection software system for industrial parts. We have also begun to look at part tolerancing in the presence of measurement error as a step toward understanding manufacture process variability from images. Both of

these topics are discussed in the paper.

1.1 Inspection and Continuous Product Improvement

The traditional model for industrial part quality assessment involves two steps; **in-process** and **post-process** or **final** inspection. Most machine vision inspection systems built in the past 30 years have been built along these lines. One problem with validating such systems is that flaw occurrence rates can be very low and hence the practical advantages of using an automated system (repeatability, accuracy, speed) is not readily realized until after a long validation period.

The notion that part geometry can be used to help guide industrial image interpretation has been around for some time. Early "geometry-driven" approaches attempted to use an actual copy of the part itself as a reference or "golden template" [Decker, 1983]. The immediate objection is that the specific part may not represent the ideal dimensions. Further, a major problem is how to interpret differences between the reference part and the part to be inspected. These differences can arise from irrelevant variations in intensity caused by illumination, uniformity or shadows. Even if the image acquisition process can be controlled, there will be unavoidable part-to-part variations which naturally arise from the manufacturing process itself, but are irrelevant to the quality of the part. Although the part reference approach has proven highly successful in the case of VLSI photolithographic mask inspection [Huang, 1983, Okamoto *et al.*, 1984] it is difficult to see how to extend this simple approach to the inspection of more complex, three-dimensional, manufactured parts without introducing some structure defining various regions and boundaries of the part geometry.

An alternative approach that became popular in the 1980's was the idea of using computer-aided-design (CAD) models to derive the necessary information to automatically program visual inspection [West *et al.*, 1991, Chen and Mulgaonkar, 1991]. The advantage of this approach is that the geometry of the object to be inspected and its tolerances can be specified by the CAD model and used to derive optimum lighting and viewing configurations as well as provide context for

the application of image analysis processes.

On the other hand, the CAD approach has not yet been broadly successful because images result from complex physical phenomena, such as specular reflection and mutual illumination. A more significant problem limiting the use of CAD models is that the actual manufactured parts may differ significantly from the idealized model. During product development a part design can change rapidly to accommodate the realities of manufacturing processes and the original CAD representation can quickly become obsolete. Finally, for curved objects, the derivation of tolerance offset surfaces is quite complex and, at the very least, requires the solution of high degree polynomial equations [Farouki, 1986].

During the 1980's, manufacturing industries attempted to introduce processes for **continuous product improvement** or **total quality**. The idea behind this was to achieve incremental improvements in product design and manufacture by monitoring and controlling critical manufacturing process parameters throughout part production and to move away from the one or two step inspection model to multiple step (i.e. near continuous) inspection. Although relatively simple, specialized vision-based monitoring devices have been successfully demonstrated, these are not often cost-effective solutions in the long run and rapidly become out-dated as manufacturing procedures change.

It is clear that the key to long term acceptance of industrial vision systems for part quality assessment is the introduction of generic methodologies that can be readily re-programmed to new but similar part geometries and can also cope with both inherent imaging distortions and part geometry variations. To address these issues we have chosen constraint templates as a medium for industrial part inspection and understanding/monitoring manufacturing process parameters together with IU standards as a way to standardize concepts and software protocols across applications, and imaging modalities (currently x-ray, ultrasound, optical).

We have currently reduced our ideas to practice in the form of an X-ray image analysis system for part inspection. This system, which is called the Image Interpretation Foundations (I²F) system is described in detail below.

2 Geometric Constraint Templates

The focus in inspection and part monitoring is on the geometry of a part. Our analysis is therefore largely based on geometry-centered representations which we call **templates**.

In our approach a template provides the context for

- monitoring geometric measurements across batches,
- shape-based material characterization,
- nominal and variational part geometry extraction (manufactured part tolerancing), and,
- intensity signal verification (flaw analysis).

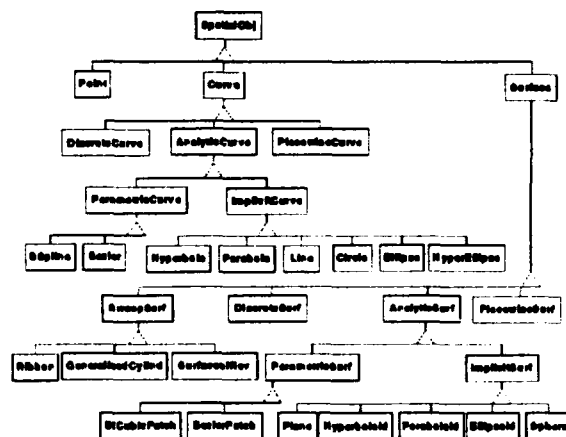


Figure 1: *Geometry hierarchy.*

We represent critical part geometry by a constraint-model which is defined by an inspector during inspection template construction. These template-defined relationships provide a flexible structure which can adapt to manufacturing process variations while at the same time insisting on the primary constraints required for a good quality part. We accommodate these part-to-part variations by solving each time for an instance of the template which satisfies all of the specified constraints, while at the same time accommodating for the observed image features which define the actual part geometry. The statistics of incidental variations can be accumulated over a large number of parts. The resulting parameter distributions can then be used to define normal ranges of part geometry. Since the template is designed directly from inspection images, the template features are consistent with both part geometry and inherent imaging distortions.

Our current geometry hierarchy is shown in figure 1 and includes 2D/3D points, as well as representations for 2D/3D curves and surfaces which are commonly used in manufacturing and design. A geometric description is built up from a set of primitive geometric classes such as **point**, **line** and **conic**.

The primitives are imbedded in a topological network which defines various connection relations between the primitives. For example, a set of line segments may be joined at common endpoints to form a polygonal chain. General geometric relations or **constraints** may be defined between the primitives such as **parallel**, **equidistant**, and **tangent continuity**. The latter condition ensures that two curve segments have equal tangents at a common intersection point. It is possible to build up quite complex and flexible models from these primitives and relations.

Except for scalar measures such as length and angle, each geometric entity is represented by a **configuration**, which is an affine transformation matrix representing the translation, rotation, and scaling from the local coordinate frame (X, Y) of the shape to the image frame (x, y) . Symbolically, in two-dimensions, a **configuration** has 3 parameters, location, orienta-

tion and size represented by 2D vectors of variables.

The **location** of a shape is described by $\mathbf{l} = (l_x, l_y)^T$. This location is usually the center or the origin of the local frame of the primitive shape.

The **orientation** of a shape in the plane is described by an angle θ , or by a unit vector $\mathbf{o} = (o_x, o_y) = (\cos \theta, \sin \theta)^T$. The latter is used to avoid trigonometric functions and use only polynomial functions of integer powers.

The **size** of a shape like an ellipse is represented by a vector having 2 scale factors, (k_x, k_y) along the major and minor axes $(a, b)^T$. To avoid division of polynomials, the inverse of the size is represented: $\mathbf{k} = (k_x, k_y)^T = (a^{-1}, b^{-1})^T$.

The description of the primitives and the geometric relationship between primitives is then represented in a uniform manner as a system of polynomials in the configuration and constraint parameters. We have found that this generic representation is adequate for many applications such as template editing, and template solving and allows new algorithms to be readily prototyped with minimal effort. In particular, the use of a uniform polynomial system for the constraints permits the development of an efficient constraint solver which is described in section 4.3.

2.1 Template Variants

Templates are defined in terms of a set of geometric entities and a set of spatial, functional, or descriptive objects. Spatial relationships between geometric objects are defined by a **constraint** (see section 2). Functional objects define the context within which geometry is used in an analysis algorithm. For example, a **checker** is used for local intensity signal analysis where a geometric primitive defines the image contour/region of interest extracted from the image. Curve-based checkers include **contour-checker** (this extracts an intensity signal along a contour), and a **hole-checker** (this enhances and extracts the signal profile along low contrast linear features). These are generic image processing representations that are also useful for building flaw-detection algorithms, (for an example see section 4.4). A **monitor** can be attached to a configuration and is used to record (i.e. monitor) a configuration parameter or dimension over image sequences. Descriptive labels can also be attached to geometric primitives, for example a **hole** to a line primitive, a **row** to a group of circles or a **cavity** to a polygon/face. This provides a higher-level interface to analysis which is more in line with the terminology used in inspection.

3 Object-Oriented Design in Machine Vision

The main advantages of object-oriented design are flexibility and code sharing. The definition of generic object classes provides standard interfaces so that new code can be quickly developed and integrated since the important data structures and variables are already in place. Two major vision systems have already been implemented along object-oriented design principles, the Cartographic Modeling Environment [Hanson and

Quam, 1988] and Power Vision [McConnell and Lawton, 1988]. The Image Understanding Environments (IUE) Project funded by DARPA [Mundy *et al.*, 1992] has also made a major contribution to this area.

The I²F System is an image understanding software system designed using object-oriented methodology and implemented using the C++ language and the X-toolkit, InterViews [Linton *et al.*, 1988]. Key features of our system include:

- The use of the **subject-view** paradigm for providing the relationships between an object, i.e. the **subject**, and the graphics display or the **view** of the object. This approach permits the development of object libraries which are not dependent on specific display mechanisms.
- Adherence to the concepts and formats defined by the PIK (Programmers Imaging Kernel) standard [ANSI, 1990]. It is expected that many image accelerator manufacturers will support the PIK standard so that the code developed in I²F will transparently run with increased throughput on a PIK standard accelerator.
- An extensive set of image feature classes which are closely integrated with geometric primitives to facilitate the geometric representation of image events.
- New object concepts to support geometric constraint programming. A hierarchy of geometric primitives and parameterized transformations of the primitives is provided to allow the description of curved shapes and to account for global geometric relations between primitives.

Over a 2 year period the I²F system has matured into a collection of libraries containing approximately 300 classes dedicated to image analysis algorithms, display utilities and interactive tools. (Figure 2). The software is divided into 2 major library groups; the IU Standards, which contains functionality like segmentation, image filtering, geometry, topology, that are common to all (i.e. not just industrial image interpretation) IU applications, and the I²F Standards which are built on top of the IU Standards and include template-specific representations and functionality. As illustrated in Figure 3, individual applications are built on top of these two library groups.

4 The I²F Inspection System

In this section we discuss the current implementation of an inspection system which focuses on X-ray image analysis of parts. A flow chart of the system is shown in Figure 4.

4.1 Template Creation

A typical example is shown in Figure 5. The general template creation process involves first specifying a set of geometric primitives and then establishing the relationships between them. In our system this is achieved using a graphical template editing tool implemented using InterViews [Linton *et al.*, 1988]. The tool allows the user to build a template composed of a selection

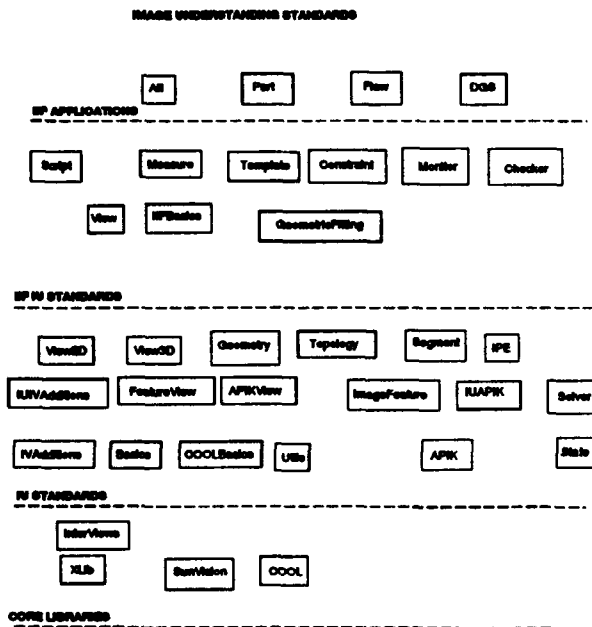


Figure 2: Core IIF Standards Libraries.

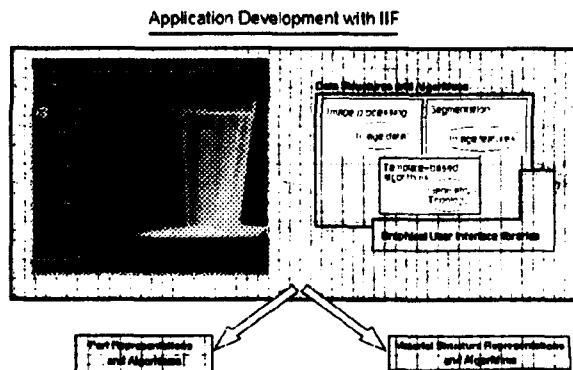


Figure 3: IIF Core software system provides the basis for a variety of template-based industrial applications.

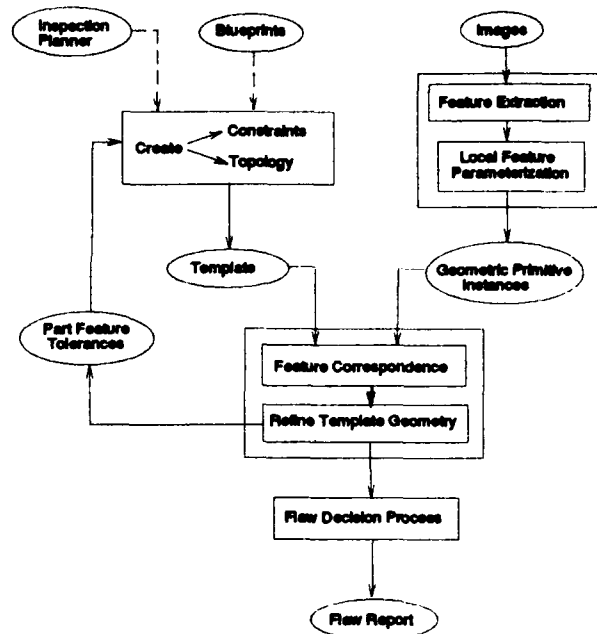


Figure 4: Flow diagram of the IIF system.

of the 4 types of geometric primitive specified in section 1.1 which are related by any of the 12 possible constraint types. A "snap-shot" view in the process of creating a template is illustrated in Figure 5a. The set of configurations in the template contains a number of lines and a number of points. These geometric entities (or rather their counterparts in the image) are subjected to a number of constraints. The template after deformation is shown in Figure 5b.

4.2 Feature Extraction and Correspondence

In the current implementation feature extraction is achieved by either a combination of morphological signal enhancement and segmentation [Noble, 1992] or the Canny edge detector [Canny, 1986]. We use eigenvectors of the feature point scatter matrix to estimate the parameters of experimental geometric primitives from the segmentation output. A feature correspondence step then performs local adjustment to register the image features with the template.

Our philosophy is to keep the local feature correspondence simple and to rely on a global template registration step to place the template in close approximation to the image features. This is done by registering on one or more geometric features on a part that are invariant to part-to-part variations. These features are determined in an experiment run on a set of good parts using a monitor template to determine registration features which do not shift between images. The geometric transformation is computed for each new image and applied to the inspection template prior to applying the inspection algorithm.

4.3 Solving Constraints

The objective of the constraint solver is to find an instance of the inspection template which satisfies all

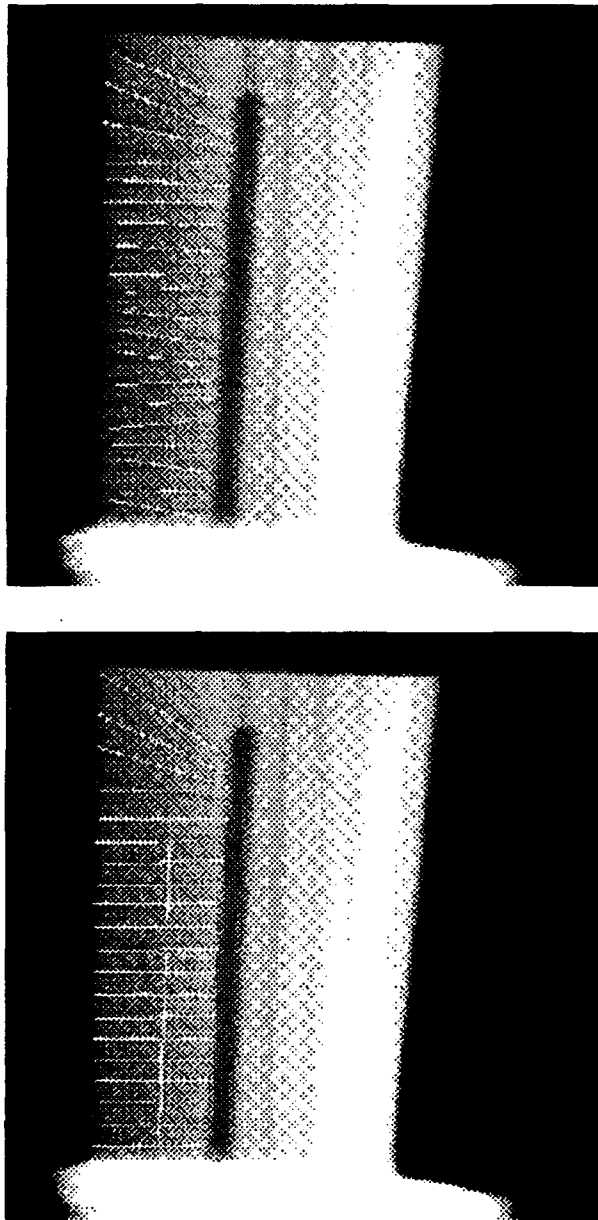


Figure 5: Template construction and solving.

of the geometric constraints defined by the template and at the same time, minimizes the mean-square error between the template primitives and the image features. The mean-square error can be expressed as a convex function of the template parameters and a geometric description of the image features.

The two goals of finding the global minimum of a convex function, $\nabla f(\mathbf{x}) = 0$, and satisfying the constraints, $\mathbf{h}(\mathbf{x}) = 0$, are combined to give a constrained minimization problem. A linear approximation to this optimization problem is:

$$\begin{cases} \nabla^2 f(\mathbf{x}) \, d\mathbf{x} = -\nabla f(\mathbf{x}), \\ \nabla \mathbf{h}(\mathbf{x}) \, d\mathbf{x} = -\mathbf{h}(\mathbf{x}). \end{cases} \quad (1)$$

Since the two goals cannot in general be simultaneously satisfied, a least-square-error satisfaction of $\nabla f(\mathbf{x}) = 0$ is sought. The constraint equations are multiplied by a factor \sqrt{c} , which determines the weight given to satisfying the constraints versus minimizing the cost function. Each iteration of (1) has a line search that minimizes the least-square-error:

$$m(\mathbf{x}) = |\nabla f(\mathbf{x})|^2 + c |\mathbf{h}(\mathbf{x})|^2 \quad (2)$$

which is a merit function similar to the objective of the standard penalty method.

Starting with $\sqrt{c} = 0$, system (1) converges to the unconstrained global minimum first, and so avoids local minima and singularities on the constraint surface. This convergence is efficient, because the fitting error $f(\mathbf{x})$ is well approximated by a quadratic, and so the Hessian $\nabla^2 f$ is almost constant. When the Hessian $\nabla^2 f$ is constant, the best-fit surface is a linear subspace with zero curvature and no singularity, a lot simpler than the constraint surface. Note that this approach is very similar to the construction of a Tikhonov stabilizing functional [Bertero *et al.*, 1988]. The convex fitting function $f(\mathbf{x})$ can be viewed as a regularizer in the solution of $d\mathbf{x}$. It makes the constraint problem well-posed by using empirical data whenever additional constraints are needed to pin down free variables in $\mathbf{h}(\mathbf{x}) = 0$. Levenberg and Marquardt have shown that varying \sqrt{c} by factors of 10 is an effective method to force convergence for nonlinear systems [Marquardt, 1963, Press *et al.*, 1988].

4.4 Part Verification and Flaw Decision-Making

The output from the constraint solver is a set of deformed primitives which can be used to further refine the parameter values and tolerances of the inspection template, or for flaw decision-making. For example, the parameters derived from the deformed primitives can be compared to the template parameters to detect geometric flaws such as inaccurate drilled hole diameters [Noble *et al.*, 1992].

The adapted inspection template primitives can also provide the context for applying specialized algorithms to characterize shape and intensity-based properties of subtle flaws. In figure 6 we illustrate one example in which a **checker** is associated with each of

3 drilled-holes. In physical terms, flaws can arise if the drilling process goes wrong and surplus material. In an x-ray image an absence of material appears as an unexplained low intensity region. Automated flaw inspection involves extracting 1D flaw signals in the direction along the hole and classifying the signal by comparing the flaw profile against the expected profile in intervals along the signal length. We are currently implementing a database of generic flaw algorithms of this kind that can be selected through an inspection algorithm editor by a user to build specialized inspection algorithms for parts with different geometries. In practice, a typical part may have to be checked for between 1 and 20 different flaw types.

5 Toward Continuous Product Improvement

By computer vision applied to continuous product improvement we mean using image-based analysis techniques to understand manufacturing process variability (how a design part differs from a manufactured part) and to implement vision systems for monitoring part manufacture on-line. To date, our research in this area has focussed on developing techniques to reliably extract manufactured part dimensions and models from images in the presence of measurement error. The issue here is that what you measure in an image is a nominal dimension + manufacturing tolerance + imaging noise. However, to verify that a part meets design specs. you need to know that a part dimension lies within manufacturing tolerance bounds. Put another way, you can not take the estimates of the parameter distributions computed from the *image* as the parameter distributions of the *part* because imaging the part has introduced some measurement error. (Note that since you are typically dealing with a 2D projection of a 3D object, a correction may also have to be made for pose variation between images. However we do not consider this problem here). So how do you distinguish between variability due to manufacturing and variability due to imaging noise? Our approach to solve this problem is based on the analysis of variance components using Bayesian techniques and is briefly described below.

5.1 Bayesian Part Tolerancing with Measurement Error

Part tolerancing is generally perceived as the problem of determining the variations in dimensions for an object where the errors originate from the manufacturing process. On the other hand measurement error analysis typically refers to the problem of quantifying the variations in measurements due to the sensing/imaging process. Although the solid modelling community is beginning to make progress in the first area [Parkinson, 1984, Light and Gossard, 1982, Turner, 1988, Requicha, 1983, Requicha and Chan, 1986, Fleming, 1988], and there has been considerable work in the latter area (though surprisingly not so in computer vision), little, if any attention has been given to tackle the problem of part tolerancing *with* measurement.

Suppose we can take a number of measurements of each part, figure 7. Manufacturing error (part toler-

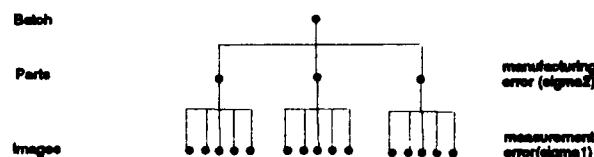


Figure 7: Error model.

ance) is added when the part is made. After imaging the part, further measurement error is introduced. If we assume an additive noise model then the j th measurement on the i th part, x_{ij} , can be modeled by:

$$x_{ij} = x + \xi_i + \eta_{ij} \quad (3)$$

Here, x is the true (unknown) dimension, ξ_i is the perturbation due to manufacturing error (tolerance) and η_{ij} is the perturbation due to measurement error. This is called a **random effect model** [Box and Tiao, 1973] and provides a statistical model for part tolerancing with measurement error. Note also that we could go one step further by adding another "process" layer to the hierarchical in figure 7 and represent batch-to-batch variations by a third variance component.

However, consider the case of two variance components, where our goal is to separate out the part tolerance (process variation) from the measurement error. It turns out that the solution to this problem for the case of a single dimension can be found by Bayesian variance component analysis of random effect models using the Gibbs sampling method [Hastings, 1970, Geman and Geman, 1984, Gelfand *et al.*, 1990, Tanner, 1991]. The Bayesian philosophy allows you to input prior knowledge (the "design" dimension) and the data samples modify this to reflect manufacturing process variability. Further, an attractive practical advantage of this approach is that it can be realized very simply with an experimental setup whereby separate images are taken of different samples of a part, figure 8. Finally, an advantage in using Gibbs sampling is that it provides a mechanism for estimating, numerically, a parameter distribution rather than just a "best" estimate for the nominal value of a parameter. This is desirable from a manufacturer's perspective as it has been shown that tolerance analysis using parameter distributions (statistical tolerancing) leads to larger allowable tolerance limits on design variables than can be achieved using upper/lower limit analysis (worst case tolerancing) [Michael and Siddall, 1981]. Hence, parts can be made using less precise (and hence cheaper) manufacturing processes provided that the probability distributions of critical part parameters can be monitored.

We have performed preliminary experiments using a Gibbs sampler for single dimension part tolerancing with measurement error and have recently started looking at variance component analysis for constraint templates. We can analyze the case of a linear constraint (eg a linear size gradient) using the Gibbs sampling solution to the Bayesian linear model [Lindley

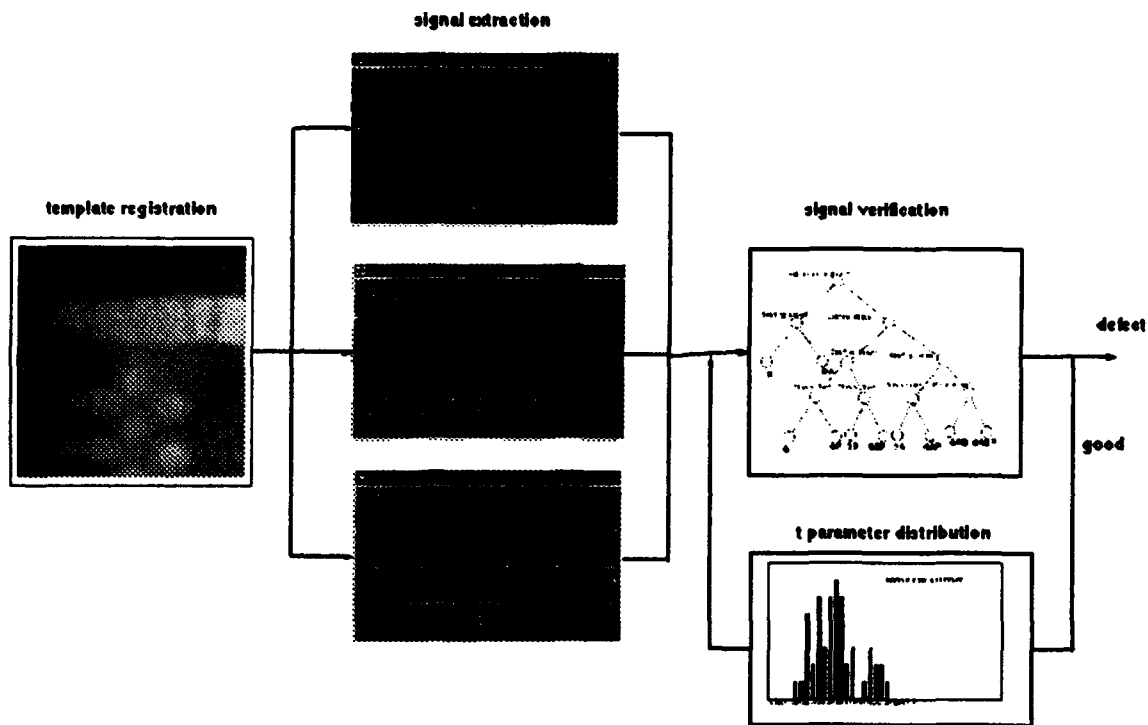


Figure 6: *Template-based flaw signal decision-making.*

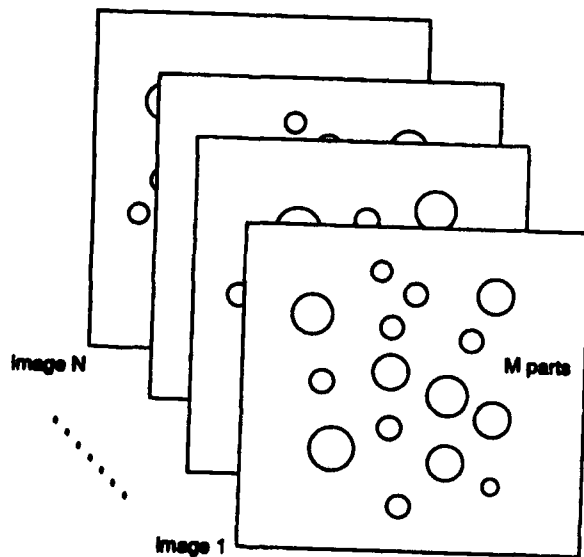


Figure 8: *Variance component analysis for a single parameter.*

and Smith, 1972]. The extension to nonlinear constraints is not so straightforward and is the subject of our current research. Our ultimate goal is to be able to derive a tolerance template from sequences of images of good parts where tolerances capture process variability and to use this to guide part verification and flaw decision-making.

6 Summary

In this paper we have presented an overview of the I²F System, an object-oriented image analysis being developed for machine vision industrial inspection and process monitoring. We have described aspects of the system design and a software inspection system for detecting flaws in parts. We have also discussed some preliminary work in the area of part tolerancing. There are many topics we plan to study further including: multi-scale feature correspondence; multivariable extensions of variance component analysis for constraint templates of parts; algorithm performance assessment; application to other modalities (ultrasound and infra-red images) and part integrity verification for 3D (volumetric) images.

Acknowledgements

The I²F team, namely Jim Farley, Van-Duc Nguyen and Ahn-Tuan Tran is responsible for the development and implementation of many of the ideas described in the paper. This work has also benefited from input by Bill Hoffman and Patti Vrobel. Jean Ponce made

useful comments on this paper.

References

- [ANSI, 1990] ANSI. *ANSI X3H3.8 - Imaging Applications Programmer Interface Task Group: Programmer's Imaging Kernel (PIK)*. Strawman V8b edition, December 1990.
- [Bertero et al., 1988] M. Bertero, T.A. Poggio, and V. Torre. Ill-posed Problems in Early Vision. *Proceedings of the IEEE*, 76(8):869-889, 1988.
- [Box and Tiao, 1973] G.E.P. Box and G.C. Tiao. *Bayesian Inference in Statistical Analysis*. Addison-Wesley, Reading, M.A., 1973.
- [Canny, 1986] J.F. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679-698, November 1986.
- [Chen and Mulgaonkar, 1991] C. Chen and P. Mulgaonkar. CAD-Based Feature-Utility Measures For Automatic Vision Programming. In *Proceedings IEEE Workshop on Automated CAD-Based Vision*, page 106, Lahaina, Hawaii, June 1991.
- [Decker, 1983] H. Decker. A Difference Technique for Automatic Inspection of Casting Parts. *Pattern Recognition Letters*, 2:125-129, 1983.
- [Farouki, 1986] R. Farouki. The approximation of non-degenerate offset surfaces. *Computer Aided Geometry Design*, 3(1):15-44, 1986.
- [Fleming, 1988] A. Fleming. Geometric Relationships between Toleranced Features. *Artificial Intelligence*, 37:403-412, 1988.
- [Gelfand et al., 1990] A.E. Gelfand, S.E. Hills, A. Racine-Poon, and A.F.M. Smith. Illustration of Bayesian Inference in Normal Data Models Using Gibbs Sampling. *Journal American Statistical Society*, 85(412):972-985, December 1990.
- [Geman and Geman, 1984] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(4):721-740, 1984.
- [Hanson and Quam, 1988] A. Hanson and L. Quam. Overview of the SRI Cartographic Modeling Environment. In *Proceeding of the DARPA Image Understanding Workshop*, pages 576-582, Cambridge, MA., April 1988.
- [Hastings, 1970] W.K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Application. *Biometrika*, 57:615-632, 1970.
- [Huang, 1983] G. Huang. A robotic alignment and inspection system for semiconductor processing. In *International Conference on Robot Vision and Sensory Control*, pages 644-652, Cambridge, MA, 1983.
- [Light and Gossard, 1982] R. Light and D.C. Gossard. Modifications of Geometric Models through Variational Geometry. *Computer Aided Design*, 14(4):209-214, 1982.
- [Lindley and Smith, 1972] D.V. Lindley and A.F.M. Smith. Bayes Estimates for the Linear Model. *Journal of the Royal Statistical Society, Series B*, 34:1-42, 1972.
- [Linton et al., 1988] M.A. Linton, P.R. Calder, and J.M. Vlissides. InterViews: A C++ Graphical Interface Toolkit. Technical Report CSL-TR-88-358, University of Stanford, Stanford, CA., July 1988.
- [Marquardt, 1963] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society of Industrial Applied Mathematics*, 11(2):431-441, 1963.
- [McConnell and Lawton, 1988] C.C. McConnell and D.T. Lawton. IU Software Environments. In *Proceeding of the DARPA Image Understanding Workshop*, pages 666-677, Cambridge, MA., April 1988.
- [Michael and Siddall, 1981] W. Michael and J.N. Siddall. The Optimization Problem with Optimal Tolerance Assignment and Full Acceptance. *Transactions of the ASME Journal of Mechanical Design*, 103:842-848, October 1981.
- [Mundy et al., 1992] J. Mundy, T. Binford, T. Boulton, A. Hanson, R. Beveridge, R. Haralick, V. Ramesh, C. Kohl, D. Lawton, Morgan, D., K. Price, and T. Strat. The Image Understanding Environments Program. In *Proceeding of the DARPA Image Understanding Workshop*, pages 185-214, San Diego, CA., January 1992.
- [Noble et al., 1992] A. Noble, V.D. Nguyen, C. Marinos, A.T. Tran, J. Farley, K. Hedengren, and J. Mundy. Template Guided Visual Inspection. In *Proceedings of the European Conference on Computer Vision*, pages 893-901, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag.
- [Noble, 1992] J. A. Noble. Finding Half Boundaries and Junctions in Images. *Image and Vision Computing*, 10(4):219-232, May 1992.
- [Okamoto et al., 1984] K. Okamoto, K. Nakahata, S. Aiuchi, M. Nomoto, Y. Hara, and T. Hamada. An automatic visual inspection system for LSI photomasks. In *Proceedings of the International Conference on Pattern Recognition*, pages 1361-1364, Montreal, Canada, 1984.
- [Parkinson, 1984] D.B. Parkinson. Tolerance of component dimensions in CAD. *Computer Aided Design*, 16(1):25-32, January 1984.
- [Press et al., 1988] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1988.

- [Requicha and Chan, 1986] A.A.G. Requicha and S.C. Chan. Representation of Geometric Features, Tolerances and Attributes in Solid Modelers Based on Constructive Geometry. *IEEE Journal of Robotics and Automation*, 2(3):156-166, September 1986.
- [Requicha, 1983] A.A.G. Requicha. Toward a Theory of Geometric Tolerancing. *International Journal of Robotics Research*, 2(4):45-60, Winter 1983.
- [Tanner, 1991] M.A. Tanner. *Tool for Statistical Inference*, volume 67. Springer-Verlag, New York, 1991.
- [Turner, 1988] J.U. Turner. New Methods for Tolerance Analysis in Solid Modelling. In *IEEE International Conference on CIM*, pages 306-314, Troy, NY, 1988.
- [West et al., 1991] A. West, T. Fernando, and P. Dew. CAD-Based Inspection: Using a Vision Cell Demonstrator. In *Proceedings IEEE Workshop on Automated CAD-Based Vision*, page 155, Lahaina, Hawaii, June 1991.

Sensor Modeling, Markov Random Fields, and Robust Localization for Recognizing Partially Occluded Objects

Mark D. Wheeler *
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
email: mdwheel@cs.cmu.edu

Katsushi Ikeuchi †
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
email: ki@cs.cmu.edu

Abstract

We describe three components of our Vision Algorithm Compiler for recognizing rigid objects in range images: realistic sensor modeling, a novel hypothesis-generation algorithm and a robust localization method.

We use *sensor modeling* to build prior models of object appearances that account for constraints due to sensor and feature-extraction algorithm characteristics in addition to model geometry. We approach hypothesis generation as a search for the most likely set of hypotheses based on our prior knowledge. The *Markov random field* (MRF) formalism and Highest Confidence First estimation provide us with an efficient and effective technique for performing this search. Our algorithm has shown the ability to recognize objects while limiting the number of hypotheses requiring verification.

Our pose refinement algorithm uses a *robust* estimator to deal with the problem of abundant outliers from our models in images due to occlusion and noise. The algorithm has been found to be much less sensitive to outliers than the least squares solution.

1 Introduction

Recognizing known objects in images is a fundamental problem in computer vision. Much work has been done on object recognition in intensity images as well as range images. Several researchers have developed model-based vision systems for object

recognition in range images [Bhanu, 1984, Bolles *et al.*, 1987, Grimson and Lozano-Perez, 1987, Ikeuchi and Kanade, 1988, Fan, 1990, Kim and Kak, 1991, Stein and Medioni, 1992].

Most model-based vision systems do not utilize realistic prior models of the appearance of the objects. This affects both efficiency and robustness of the recognition algorithm. Without an accurate model of sensor and segmentation characteristics, the hypothesis-selection procedure must compensate for the inaccuracies by loosening the constraints and, thus, increasing the number of incorrect hypotheses that are generated. Our solution is to use *sensor modeling* to build accurate prior models of constraints due to sensor and segmentation characteristics in addition to model geometry.

A requirement that image features be separated into groups belonging to single objects is a weak point common to many recognition systems. This grouping operation is not in general possible with a purely data driven segmentation algorithm. Our algorithm does not require that image features are segmented into groups belonging to single objects.

When unknown objects are present in the image, the performance of most systems degrades drastically. A recognition system should not expend precious resources on unlikely possibilities. Our hypothesis-generation algorithm is intended to reduce the number of hypotheses requiring verification by accurate *selection* of hypotheses and *optimal ordering* of hypotheses for verification. We formulate the task of "optimal" hypothesis selection as a search for the most likely set of matches based on our prior knowledge. This is accomplished by integrating observed image features and our prior knowledge (constraints) in the formalism of Markov random fields (MRF). With the MRF formulation, our search for the most likely hypotheses is phrased as a *maximum a posteriori* (MAP) estimate of the MRF. The number of verifications is further reduced by ordering the hypotheses in terms of their likelihood based on our prior knowledge.

The effects of partial occlusion are not explicitly

*Supported by a National Science Foundation Graduate Fellowship.

†This research was sponsored by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the U.S. Government, or the National Science Foundation.

modeled by most recognition systems. The assumption that enough unoccluded features will be visible to perform recognition/localization is not always true. Our localization algorithm explicitly models the effects of partial occlusion by using an error distribution that is relatively insensitive (compared to least-squares approaches) to outliers due to occluded features or noise.

In this paper, we present three components of our recognition system which take steps to reduce or eliminate the above problems. We present the sensor-modeling approach to generating realistic constraints for object recognition, an efficient hypothesis-generation algorithm, and a robust method for localizing hypothesized models in range images. Sensor modeling, hypothesis generation utilizing MRFs, and localization are key elements of our Vision Algorithm Compiler (VAC) for object recognition [Wheeler and Ikeuchi, 1992]. Our current system is designed to recognize polyhedral models in range images. There are three distinct components of our VAC system: user-defined modules, the compiler, and the executable recognition program. The user-defined modules consist of object models, sensor models, image processing and segmentation modules, and the feature modules. The VAC uses the user-defined modules and sensor modeling to generate the recognition program for the specified models and sensors. The compiled prior models are integrated with the hypothesis-generation and localization algorithms to form the executable recognition program.

In section 2, we describe the sensor-modeling process for generating realistic prior models of the object's appearance. We present our MRF formulation for hypothesis generation in Section 3. Section 4 describes our robust localization algorithm. In section 5, we present some results from recognition and localization experiments. Section 6 summarizes the advantages of our methods.

2 Sensor Modeling

The constraints used by the hypothesis-generation process of most model-based vision systems are based solely on the geometric models of the objects and do not account for sensor or feature-extraction characteristics. Without an accurate model of sensor and segmentation characteristics, the hypothesis-selection procedure must compensate for the inaccuracies by loosening the constraints and, thus, increasing the number of incorrect hypotheses that are generated. Our solution is to use *sensor modeling* to build accurate prior models of constraints due to sensor and segmentation characteristics in addition to model geometry. The inclusion of imaging and processing effects is the essential difference between our prior models and the view-variation distributions used by [Burns and Riseman, 1992].

Our current system's sensor modality is range data and our low-level vision routines supply us with segmented planar surfaces. In our application, a hypothesis is a match between a planar region R_i of the image and model face M_j , and each region R_i is described by a vector of feature values $\vec{f}_{ri} = (f_{ri}^1, f_{ri}^2, \dots, f_{ri}^n)$. The features for this system are specified over 3-D surfaces corresponding to planar regions extracted from images. Our first-order features include region area, maximum second moment, minimum second moment, and maximum axis length. Second-order features include simultaneous visibility, relative orientation, and maximum distance between surfaces.

The constraints used by our hypothesis generation algorithm are in the form of probability distributions of the appearance of model faces represented by conditional distributions $P(f_{ri}^n | M_j)$. The sample range images are generated using an appearance simulator developed by [Fujiwara *et al.*, 1991]. We can use these sample segmented images to compute a prior distribution $P(f_{ri}^n | M_j)$ for each feature and each model face M_j . The prior distributions are approximated by generating many sample images (320 images) of our object models and segmenting the images using our low-level vision routines. The simulated images are segmented and the features of each image region are calculated. Figure 1 shows an example iteration of this process. In this work, the viewing directions are "uniformly" distributed on the unit sphere; however, it would be easy to modify the distribution to reflect real world constraints and biases for particular objects (i.e., the bottom of a stapler is rarely visible).

Since this is a simulation, we know the correspondence between the model surfaces and the segmented regions. Thus, we can build a list of the sampled feature values for each model surface. The feature values for each model face are tabulated and used to form the prior distribution. Figure 2 shows a sample distribution of a model face's area value as computed using our sensor model. The simulated distributions are not normally distributed, and they are biased due to inherent characteristics of the segmentation algorithm. Additional bias occurs from self occlusion when viewing some object from certain directions. There are secondary modes corresponding to oversegmentations, where a single object surface is segmented into multiple regions. The sensor-modeling approach builds models of the information that the recognition algorithm will actually have available when viewing a known object. This information is dependent on the imaging process and segmentation algorithm, in addition to the known geometric characteristics of the model. An additional benefit of this approach is that model surfaces that are, because of geometric properties of the object, not detectable by the segmentation program will not affect the hypothesis generation.

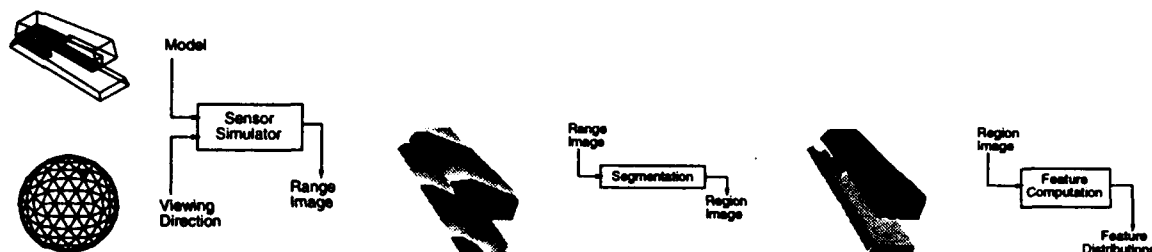


Figure 1: Generation of feature distributions. An object model and viewing direction are selected. The simulator is used to produce a range image of the object which is then segmented into regions which are used to compute the feature distributions.

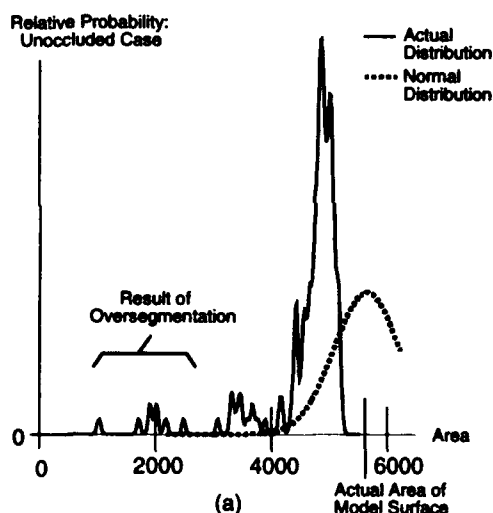


Figure 2: An example distribution of a given feature value (area) over a model face. The distribution was generated by sampling resulting area values from synthetic images of views of the object. A normal distribution centered around the actual model area value is shown to demonstrate the difference between the usual assumption of performance and the actual performance of the segmentation program.

3 Hypothesis Generation

Given a set of primitive features (i.e., planar surfaces or regions) extracted from the input image by a feature-extraction algorithm (i.e., segmentation or edge detection), the hypothesis generation procedure produces a set of *possible* model primitive to image primitive matches (hence referred to simply as hypotheses). Optimally, the generated hypotheses include all of the correct correspondences and exclude as many incorrect hypotheses as possible. To exclude incorrect matches, we must apply constraints derived from our prior knowledge.

We are considering many hypotheses simultaneously and wish to choose the most likely subset of these. We can think of the hypotheses as forming a random field of variables each of which can be assigned a discrete value of **on** or **off**. A hypothesis labeled **on** indicates that the hypothesis is assumed to be correct.

The hypotheses display Markovian characteristics. For example, if two hypotheses provide mutual support for each other, and one of them is correct, it is more likely that the other is correct. A similar dependency exists between contradicting hypotheses. These dependencies can be thought of in terms of conditionally dependent probability distributions.

3.1 Formulation of Hypothesis Generation using Markov Random Fields

MRFs are used to represent the probability distribution of values, ω_i , of a set of random variables, X_i , each of which may be conditionally dependent on a set of *neighbor* variables. Given a set of independent observations, O_i , the most likely state of the MRF variables can be found by minimizing its posterior energy function

$$U(\omega|O) = \sum_{c \in C} V_c(\omega) - T \sum_i \log P(O_i|\omega_i) \quad (1)$$

where C is the set of cliques of related (neighbor) variables in the MRF. The posterior distribution is

in terms of things we may be able to calculate or specify: clique potentials $V_c(\omega)$ (represent higher-order, prior constraints among related variables) and prior distributions for our observations $P(O_i|\omega_i)$.

By representing our hypothesis space using a MRF, we can formulate the search for the most likely hypotheses as a *maximum a posteriori* (MAP) estimate of the MRF. For a review of MRFs and their applications to computer vision, we refer the reader the descriptions found in [Chou and Brown, 1990, Cooper, 1989].

We phrase our search for the most likely hypotheses as a MAP estimation problem by defining our constraints in terms of clique potentials and likelihoods in the MRF framework. With this formulation, we can apply a MAP estimation procedure to our MRF with the result being the set of hypotheses with the highest probability of occurring based on our prior knowledge. Each variable in the MRF represents a match hypothesis, (R_i, M_j) , between region R_i and model face M_j . The variables can be labeled either **on** or **off** indicating our belief or disbelief in the hypothesis.

The i th region is described by a vector of feature values $\vec{f}_{ri} = (f_{ri}^1, f_{ri}^2, \dots, f_{ri}^n)$. For computational reasons, we assume that these features are independent for a given model face. If the features are not independent, then we have redundant features which are not providing new information and should be removed. The independence assumption gives us:

$$\log P(\vec{f}_{ri}|M_j) = \sum_n \log P(f_{ri}^n|M_j). \quad (2)$$

We need to determine the likelihood that an image region, R_i , arose from the presence of a model face, M_j , in the scene. This is the probability of observing R_i assuming that the match hypothesis (R_i, M_j) is correct. We model this using a prior distribution. In terms of our label set, we equate

$$P(R_i|(R_i, M_j) = ON) = P(R_i|M_j) = P(\vec{f}_{ri}|M_j), \quad (3)$$

and we can easily calculate it with equation 2. To calculate the likelihood that a hypothesis (R_i, M_j) is incorrect, we equate this to the likelihood that R_i actually arose from any of the other model faces:

$$P(R_i|(R_i, M_j) = OFF) = \sum_{k \neq j} P(R_i|M_k) = \sum_{k \neq j} P(\vec{f}_{ri}|M_k). \quad (4)$$

Equations 3 and 4 provide us with the prior probabilities of the observations $P(O_i|\omega_i)$ required for the posterior energy function of Equation 1. Next, we need to specify the clique potentials which first requires a definition of the neighborhoods of the hypotheses (variables).

The two neighborhoods over the hypotheses are N^+ for supporting hypotheses and N^- for contradictory hypotheses. The rule that determines the N^- neighborhood is:

$$\forall R_i, M_m, M_n \neq M_m \quad ((R_i, M_m), (R_i, M_n)) \in N^- \quad (5)$$

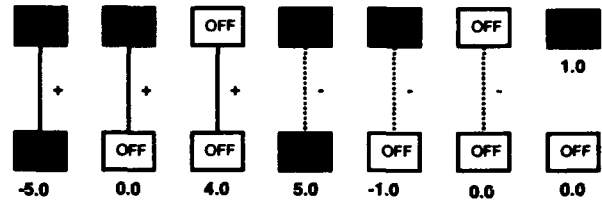


Figure 3: Clique potentials for the six possible configurations of hypothesis labels and neighbor types in 2-cliques and the two possible 1-cliques.

The above rules essentially state that hypotheses corresponding to the same region are contradictory—we would like one hypothesis per region. The rule that determines the N^+ neighborhood is:

$$\begin{aligned} &\forall R_i, R_j \neq R_i, M_m, M_n \neq M_m \\ &(\text{model}(M_m) = \text{model}(M_n)) \wedge \\ &\text{consistent}((R_i, M_m), (R_j, M_n)) \\ &\Rightarrow ((R_i, M_m), (R_j, M_n)) \in N^+ \end{aligned} \quad (6)$$

where $\text{model}(M_m)$ is the model in the model base to which the face M_m belongs, and $\text{consistent}()$ determines whether the two hypotheses are spatially and geometrically consistent based on the relational features. This rule specifies that if two hypotheses are consistent with respect to our prior constraints then they provide mutual support for each other.

If it is possible to group regions into sets belonging to the same object, a neighborhood specification rule can easily be added to enforce the required constraints. We do not use this constraint since we do not assume that grouping regions belonging to the same object is possible from a purely data-driven approach to segmentation.

For efficiency concerns, we limit our energy function to 1-cliques and 2-cliques. The clique potentials (corresponding to $V_c(\omega)$ in equation 1) used in our experiments appear in Figure 3. For example, the first clique in the figure shows that when the hypothesis is **on** and a consistent (N^+) neighbor hypothesis is **on**, -5.0 is the potential of that 2-clique. The potentials were determined experimentally to conform with our sense of consistency and mutual support among hypotheses; of course, a systematic method would be preferred. We are able to compute distributions over relations between regions, but at this point have not integrated these distributions into our formulation. Instead, we generate thresholds from these distributions to compute the relation $\text{consistent}()$.

We can now construct a MRF and search for the most likely hypotheses. To help the reader visualize a typical resulting MRF, a very simple example is shown in figure 4. This is an example MRF constructed for a model base containing two similar geometric models and an image of the first model. In this case, a hypothesis is generated for all pairs of regions R_i and model faces M_j that have nonzero

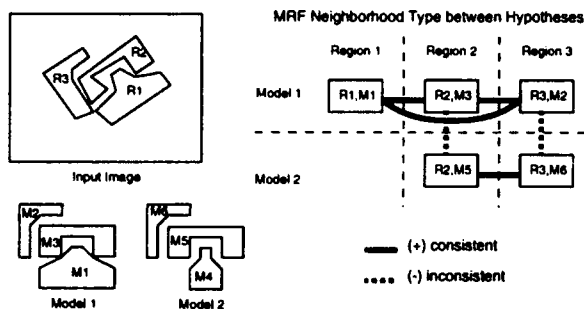


Figure 4: An example MRF produced from a simple scene containing two regions with a model base containing a tetrahedron.

conditional probabilities $P(R_i|M_j)$. The neighborhood relation of these hypotheses is simply that hypotheses for the same region are inconsistent, and hypotheses for the same model are consistent.

3.2 Highest Confidence First Search

At run time, the recognition program segments the image and computes the first-order features over all regions and relational features over all pairs of regions. The segmentation algorithm used here is the same as used in the sensor modeling phase. The segmented regions are used to build our MRF, which represents our hypotheses and prior constraints. Using equations 2 and 4, we first compute the log-likelihoods of the observation of R_i , given that hypothesis (R_i, M_m) is correct, $\log P(R_i|(R_i, M_m) = ON)$, and incorrect, $\log P(R_i|(R_i, M_m) = OFF)$. When computing $P(\tilde{f}_{ri}|M_m)$, we use a lower bound of the probability to cut off the computation and essentially throw away highly unlikely hypotheses. This lower bound is chosen such that reasonable hypotheses are not thrown away and seems to have very little effect on the final results, while reducing the size of the MRF considerably. With the thresholded hypotheses, we then determine the neighborhood systems N^+ and N^- using the rules specified in Equation 6 and 5.

Once the MRF is created, we wish to find the most likely set of hypotheses based on the constraints of the image. We use an estimation procedure called Highest Confidence First (HCF) developed by [Chou and Brown, 1990]. HCF is a sub-optimal estimation procedure but was chosen because of its efficiency and evidence of good performance in other applications [Chou and Brown, 1990, Cooper, 1989]. HCF performs a steepest-descent search in an augmented state space. The MRF variables are placed in a heap ordered by the confidence in the variable's current value. The confidence is defined as the energy difference between the current and best value. The variable at the top of the heap has its value changed to the best possible (in the current state of the MRF). The confidence values of

the first variable and its neighbor variables are recalculated, and the heap is adjusted. The use of confidence values essentially forces the algorithm to start with variables where a value is "most obvious" or has the least competition among possible values. The behavior of the HCF search for the most likely set of hypotheses is very similar to the idea of the "focus feature" method of [Bolles *et al.*, 1987]. When there are obvious matches available, the HCF search dives in by turning on the most obvious match first. This creates a ripple effect for matches consistent with obvious matches. Given a MRF with N variables, the HCF algorithm takes $O(N \log N)$ to initially create the heap and $O(\log N)$ to adjust the heap after modifying a variable's value, assuming the size of the neighborhoods is constant. In practice, [Chou and Brown, 1990] found that the variables are modified slightly more than once on average (consistent with our experience in this application) giving an $O(N \log N)$ performance.

After HCF estimation is completed, the hypotheses labeled on are considered for verification. From the results of HCF, we can create a list of consistent cliques (of order 3 and lower) of matches using the active hypotheses and their neighbor relations. The verification phase must determine which of these hypotheses describe objects that are in the scene.

A successful verification of a hypothesis eliminates other competing hypotheses from consideration. Therefore, a good ordering of the hypotheses for verification can reduce the number of hypotheses requiring verification. Traditionally, hypotheses are ordered based on the saliency (discrimination ability) or size of the image features. These heuristics have proven useful for hypothesis ordering; however, to minimize verifications, we want to first verify the most likely hypotheses—not necessarily those with the most salient or largest image features. We order the hypothesis cliques by the average of the likelihood ratios, $P(R_i|(R_i, M_j) = ON)/P(R_i|(R_i, M_j) = OFF)$ (see section 3.1), of their constituent match hypotheses—checking the most likely first. Thus, our hypothesis-generation method is "optimal" in terms of ordering for verification with respect to our prior knowledge.

4 Localization

Several factors exacerbate the localization problem: we may not have enough constraints from our matches to determine the location of the model accurately, inaccuracies in our region data due to noise and partial occlusion will lead to errors in location estimates, and our objects may vary slightly from the models causing errors in alignment along edges and surfaces. Using primitive matches alone, we are able to get crude estimates of rigid transformation parameters. Thus, localization based on our matches is assumed to be inaccurate but can serve as a good starting point for a local search for the

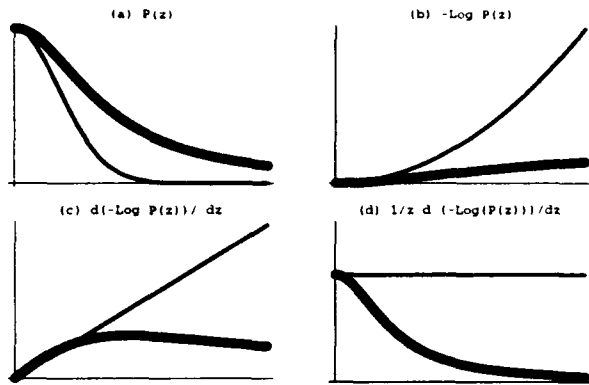


Figure 5: Comparison of Gaussian and Lorentzian distributions and their effect on outliers. The Lorentzian is in bold. (a) Gaussian and Lorentzian distributions, (b) $\rho(z)$ of the Gaussian and Lorentzian distributions, (c) the derivative of $\rho(z)$ which is the magnitude of the “force” corresponding to the data error, (d) the weight of the error vector as a function of the error magnitude for Gaussian and Lorentzian distributions.

best set of model parameters.

We can define a parameterized template to model our object and specify an energy function over the model parameters which relates how closely the model matches the image data. Then, we can perform a search over the parameter space to find the best parameters by minimizing the energy function. Since we are dealing with 3-D images, we define the template of a model to be a set of points sampled from the surface of the model. Our constraint on the templates is that visible points on the model surface match range data points in the image. The template of a rigid model is parameterized by rigid-body transformation parameters (rotation and translation).

We assume that parts of the object surfaces are often occluded. Occlusion can be due to self-occlusion, nearby objects in the scene, and even sensor shadows (visible portions of the scene which don't receive light from the light-striper in a light-stripe range finders). Occluded points are considered to be outliers as are noisy points due to illumination irregularities and sensor error. If outliers are likely, a least-squared-error estimation procedure is not desirable since the estimated parameters will be affected more by noise than the actual data. The shape of the error distribution determines how likely outliers are assumed to occur. Least-squares estimates are very sensitive to outliers since all errors are equally weighted proportional to their magnitude. Instead, we would like an estimation procedure which throws out (or gives low weight to) the true outliers. This simply corresponds to a MAP estimate using a distribution where large errors are more likely than in a normal distribution.

In this work, we use a *Lorentzian* distribution

$$P\left(\frac{z}{\sigma}\right) \propto \frac{1}{1 + \frac{1}{2}\left(\frac{z}{\sigma}\right)^2}$$

to perform the MAP estimate of our model parameters. The Lorentzian is similar in shape to the Gaussian distribution, but the tail of the distribution is much larger indicating that outliers are assumed to occur with a higher (relative) probability than in the Gaussian noise model. Figure 5 compares the (unnormalized) Gaussian distribution with the Lorentzian distribution. The important graph is Figure 5(d) which shows the weighting (relative to magnitude) of the error vectors under the Gaussian and Lorentzian distributions. The effect of the Lorentzian is to eventually give zero weight to the true outliers hence improving the estimation of parameters. It is thus, in some sense, providing robust estimate of parameters.

The goal is to improve our model parameter estimate using the range data of our image. We define q to be the vector of model parameters (rigid body translation and rotation parameters). Using

$$\rho\left(\frac{z}{\sigma}\right) = -\log P\left(\frac{z}{\sigma}\right) = \log\left(1 + \frac{1}{2}\left(\frac{z}{\sigma}\right)^2\right), \quad (7)$$

we can find the MAP estimate of $P(q) = \prod_i P\left(\frac{z_i(q)}{\sigma}\right)$ by minimizing the energy function

$$E(q) = \sum_{i \in V(q)} \rho\left(\frac{z_i(q)}{\sigma}\right) \quad (8)$$

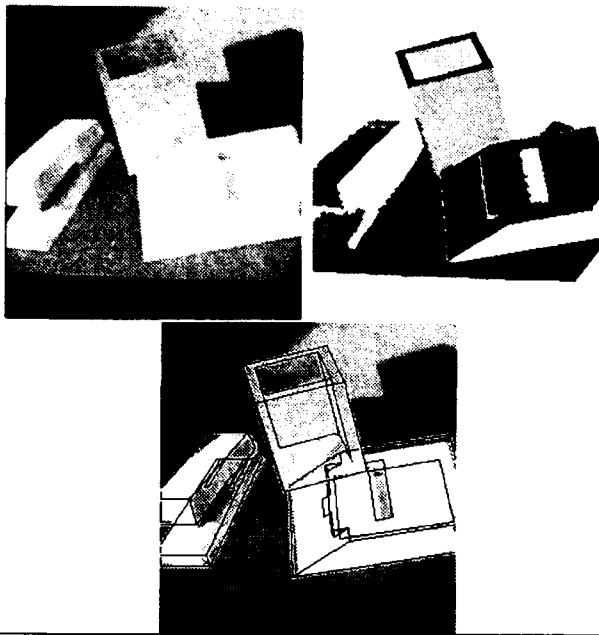
where $V(q)$ is the set of visible model points for the given model parameters q , $z_i(q)$ is the error of the i th model point given the model parameters q , and σ is the normalizing factor for the Lorentzian function which specifies the width of the distribution.

We define the error to be the distance between the model point and the data point nearest the model point:

$$z_i(q) = \min_{\vec{a} \in D} \|\vec{x}_i(q) - \vec{a}\| \quad (9)$$

where D is the set of three dimensional data points in the image, and $\vec{x}_i(q)$ is the world coordinate of the i th model point transformed using the model parameters q . The calculation of the nearest data point \vec{a} is optimized by using a k -dimensional nearest-neighbor search [Friedman *et al.*, 1977].

We use aspect information computed offline to define an efficient approximation of $V(q)$ for determining the visibility of each point. With the definition of the energy E , we utilize a form of gradient-descent search to minimize the energy. In order to reduce the effect of discontinuities in E produced from using sampled aspects to determine model point visibility, we perform all gradient-descent line searches utilizing the same set of visible points for the entire line search. Thus, the energy function is kept smooth throughout the line search.



Regions	Possible Hypotheses	Selected Cliques	Verified Cliques	Result
16	2672	48	14	penbox note-disp stapler

Figure 6: Example recognition results: intensity image (left), segmented regions (middle), wire frame overlay of models (right). The table lists the number of hypotheses generated and verified for each iteration of the algorithm using the specified occlusion parameter for computing the log-likelihoods.

5 Recognition Results and Experiments

To evaluate the performance of our hypothesis-generation algorithm, we are interested in the number of hypotheses requiring verification since the verification stage requires localization which is the expensive component of our algorithm. Experiments were conducted using a model-base of 8 polyhedral objects including a stapler, hole-cube, rolodex, castle, tape dispenser, stick, note dispenser, and pencil box.

On tested images containing known models, the correct hypotheses were consistently high in the list of selected hypotheses ordered for verification. When this occurs, the number of hypotheses verified is greatly reduced. After the known objects are recognized, all that is left are hypotheses for regions that do not correspond to known objects. Unfortunately, we must verify all of the hypotheses generated for these regions since the verification will not succeed.

Figure 6 shows the recognition results of our algorithm on a sample image. This example is typical of other tests and demonstrates the ability of our hypothesis generation to select accurate hypotheses

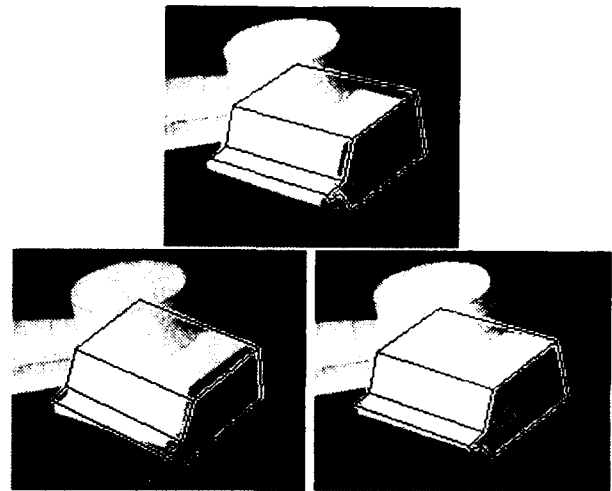


Figure 7: Comparison of the results of the least-squares formulation and the robust formulation of the error distribution on three localization problems: initial model location (top), final localized model location using least-squares (left), and final localized model location using robust formulation (right).

while limiting the size of the search. It also shows that our method can function in spite of slight partial occlusions and segmentation "errors" (see the stapler).

We performed a test on an image containing no known models. The algorithm greedily selected 107 hypothesis cliques that it thought were likely enough to justify verifying out of an absolute worst case of $\approx 2338^3$. In the case of no known models in an image, each verification fails, which means that all hypothesis cliques generated by the hypothesis generation phase must be verified. "Greedy" hypothesis generation is a beneficial attribute because almost every scene will most likely contain unknown objects, and the performance of the system will degrade quickly if the system attempts to verify every possibility. For all of the images tested, model symmetry is the major source of unnecessary hypothesis verification.

We performed tests to compare the performance of our localization algorithm using both the Gaussian and Lorentzian distributions. Figure 7 compares their solutions for an example localization task. The least-squares solution is noticeably occlusion sensitive while the results using the Lorentzian distribution are relatively insensitive to occlusion. In experiments, we have found that the initial (rough) location estimates can be perturbed by a few cm in translation and around 20 degrees in rotation without affecting the resulting solution.

Our prototype recognition program was implemented in Common Lisp for a Sun 4 workstation. The approximate execution time was 2 minutes for the segmentation, 2-5 minutes for building the MRF.

and 2-5 seconds to perform HCF and order the hypotheses for verification. The prototype of the localization procedure takes approximately 1-3 minutes per localization. We have not concentrated on making the implementation efficient. Instead, we have opted for fast a development environment in which to test our ideas.

6 Conclusions

We have introduced the use of sensor modeling for hypothesis generation for object recognition. The sensor-modeling approach has the following advantages:

- it provides realistic (accurate) constraints for "optimal" hypothesis generation by explicitly modeling the effects of the sensor, the segmentation algorithm, the geometry of the objects (including self-occlusion), and feature detectability,
- it can be used to model the effect of partial occlusion through simulation,
- it builds prior models that are robust with respect to segmentation capabilities, and
- real world constraints about likely viewing directions for particular objects can be utilized by the sensor-modeling approach to improve the hypothesis generation performance.

The MRF formalism combined with sensor modeling provides a framework for "optimal" hypothesis generation with respect to the prior knowledge from our sensor model. HCF estimation provides an efficient and effective method of performing the estimation over our MRF. Our algorithm does not require that image features are grouped into sets belonging to single objects. In experiments, our hypothesis generation algorithm has demonstrated the ability to reduce the number of hypotheses requiring verification by accurately selecting hypotheses and "optimally" ordering the hypotheses for verification.

Our localization algorithm is driven by the range data and does not rely on matches between high-level image features and model features. The pose estimate is refined through energy minimization in a manner similar to deformable templates, active contours, and snakes [Kass *et al.*, 1987]. The novelty of our algorithm is the use of a robust estimator for the energy function being minimized. The robust estimator is insensitive (relative to least-squares approaches) to outliers occurring due to partial occlusion of the object as well as other effects of complicated scenes.

References

[Bhanu, 1984] Bir Bhanu. Representation and shape matching of 3-d objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(3), 1984.

- [Bolles *et al.*, 1987] Robert C. Bolles, Patrice Horaud, and Marsha Jo Hannah. 3DPO: A three-dimensional part orientation system. In Martin A. Fischler and Oscar Firschein, editors, *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, pages 355-359. Morgan Kaufmann, 1987.
- [Burns and Riseman, 1992] J. Brian Burns and Edward M. Riseman. Matching complex images to multiple 3d objects using view description networks. In *Proceedings of Computer Vision and Pattern Recognition*, pages 328-334. IEEE, 1992.
- [Chou and Brown, 1990] Paul B. Chou and Christopher M. Brown. The theory and practice of Bayesian image labeling. *International Journal of Computer Vision*, 4:185-210, 1990.
- [Cooper, 1989] Paul Cooper. *Parallel Object Recognition from Structure (The Tinkertoy Project)*. PhD thesis, Department of Computer Science, University of Rochester, 1989. Technical Report 301.
- [Fan, 1990] Ting-Jun Fan. *Describing and Recognizing 3-D Objects Using Surface Properties*. Springer-Verlag, New York, 1990.
- [Friedman *et al.*, 1977] J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209-226, 1977.
- [Fujiwara *et al.*, 1991] Yoshimasa Fujiwara, Shree Nayar, and Katsushi Ikeuchi. Appearance simulator for computer vision research. Technical Report CMU-RI-TR-91-16, Carnegie Mellon University, 1991.
- [Grimson and Lozano-Perez, 1987] W. Eric L. Grimson and Tomas Lozano-Perez. Localizing overlapping parts by searching the interpretation tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):469-482, 1987.
- [Ikeuchi and Kanade, 1988] Katsushi Ikeuchi and Takeo Kanade. Automatic generation of object recognition programs. *Proceedings of IEEE Special Issue on Computer Vision*, 76:1016-1035, 1988.
- [Kass *et al.*, 1987] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 2(1):322-331, 1987.
- [Kim and Kak, 1991] Whoi-Yul Kim and Avinash C. Kak. 3-d object recognition using bipartite matching embedded in discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):224-251, 1991.
- [Stein and Medioni, 1992] Fridtjof Stein and Gerard Medioni. Structural indexing: Efficient 3-d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):125-145, 1992.
- [Wheeler and Ikeuchi, 1992] Mark D. Wheeler and Katsushi Ikeuchi. Towards a vision algorithm compiler for recognition of partially occluded 3-d objects. Technical Report CMU-CS-92-185, Carnegie Mellon University, 1992.

Quasi-Invariants: Theory and Exploitation

Thomas O. Binford, Tod S. Levitt *

Robotics Laboratory, Computer Science Department,
Stanford University, Stanford, CA 94305, U.S.A.

Abstract

Geometric invariants and quasi-invariants have become effective methods for real vision problems. Quasi-invariants extend invariants to widespread occurrence in computer vision. This paper presents analytic results for quasi-invariants and practical methods for their use. The mathematical definition of quasi-invariants has been used for more than 20 years. We introduce a number of quasi-invariants and prove their quasi-invariance. We compare the taxonomy of invariants with that of quasi-invariants. Several common quasi-invariants are nearly invariant.

The intent in using quasi-invariants is make approximate measurements of a class of Euclidean invariants, 3-space shape parameters. We discuss examples of the use of invariants and quasi-invariants in computer vision, in grouping and hypothesis generation for recognition. Generalized cylinders (GCs) express powerful quasi-invariants for grouping image features that define a mechanism for discrimination of GC parts, figure-ground discrimination. Quasi-invariants permit estimates of shapes of GC parts and objects formed of GC parts that enable generation of hypotheses that are verified by globally coherent interpretation by a Bayes net for evidential inference.

Quasi-invariants are inexact; a quantitative probabilistic interpretation enables efficient use of quasi-invariants in recognition in a paradigm of hypothesis generation and verification. For a majority of cases, quasi-invariant observables are approximations to corresponding body measurements in space; those quasi-invariants generate accurate hypotheses. Distributions of deviations are known or can be derived for

use in evidential inference. For a minority of cases, quasi-invariant observables give bad estimates of corresponding body measurements; those quasi-invariants generate false hypotheses that are rejected by a verification phase, in most cases at low cost. This hypothesis generation and verification mechanism depends on making some accurate measurements and some accurate hypotheses; the mechanism tolerates local errors. Computational complexity can be low. Limited completeness results are shown.

1: INTRODUCTION

Geometric quasi-invariants are one of the oldest threads in computer vision, dating to the late 1960s. Binford introduced them to extend geometric invariants in shape perception. The intent and use of quasi-invariants is somewhat different from that of invariants.

Geometric invariants have become a major topic in computer vision, beginning about 1980. Invariants have a long tradition in mathematics; many of the classic results in algebraic invariants date from the last century. Invariants have potential advantage in computational complexity over view-sensitive methods, e.g. aspect graphs. Invariants enable view-insensitive classification of objects. Aspect graph methods have very high computational complexity. Both aspect graphs and invariants accommodate very little variability in object class.

Invariants computed on one view of an object can be used as a key to index into a structured database of object models. A number of important invariants can be measured well from experimental image data. Where they exist, invariants have great utility. In many cases, there are no invariants available. Invariants are known for special cases: e.g. 4 or more points on a line; 5 or more points in the plane; algebraic curves in the plane; algebraic surfaces in 3-space; multiple views of objects. There are known to be no invariants for single views of general 3d point sets.

Invariants are valuable in important special cases. Their limitations are: 1. There are no invariants for many important cases; invariants are relatively few in

*This research was supported in part by a contract from the Air Force, F30602-92-C-0105 through RADC from DARPA SISTO, "Model-based Recognition of Objects in Complex Scenes: Spatial Organization and Hypothesis Generation".

practical vision. 2. Invariants are useful for individual objects identical to models in the database; they do not tolerate variation, e.g. stores on aircraft or open tank hatch, i.e. articulation. 3. Current use of invariants is computationally expensive, e.g. computing and indexing two invariants for all combinations of 5 points in the plane.

The pose estimation problem is to identify classes of objects from known classes and their poses from images of scenes of objects; an object class is composed of identical objects without variation, i.e. no surface marks, no articulation, no shape variation. Typical pose estimation methods are practical only with few objects.

Pose estimation is a very limited problem compared to biological vision. The generic interpretation problem is to discriminate objects, identify their classes and estimate their shapes and their poses as completely as possible with available information; object classes are composed of objects with considerable variation, e.g. humans and trees; some object classes may not be known.

View-sensitive methods for pose estimation match a dense set of views of all objects with all combinations of image features. The computational complexity of aspect graph methods is proportional to the number of combinations of image features times the number of views summed over all objects. One great contribution to computational complexity in recognition by view-sensitive methods is matching the number of views of objects. The dominant contribution to computational complexity is matching all combinations of image features. Scene complexity dominates over the number of views of objects. Both aspect graph methods and invariant methods introduce simple grouping or ad hoc grouping to reduce combinatorics of sets of image features. This work provides principled mechanisms for grouping.

Each view is a different object in the aspect graph paradigm. There are no object classes, i.e. all objects in a class are identical: there are no relations among objects other than name. E.g., the class "cars" has no meaning other than a name that includes individual cars. There is no idea of similarity. Invariants, where available, reduce this complexity; they are view-invariant and enable indexing for object.

Grouping to avoid matching all combinations of image features is the figure-ground discrimination problem. From the point of view of computational complexity, figure-ground discrimination (grouping) is a central problem in recognition. An effective mechanism for figure-ground discrimination distinguishes few possible objects among many possible combinations of image features with low complexity. Grouping typically refers to ad hoc methods for figure-ground discrimination.

Avoiding matching all views of all objects is the hypothesis generation and indexing problem. Effective hypothesis generation and indexing generates few hypotheses including the correct set of hypotheses, i.e. few views of few objects. Hypothesis generation and indexing is a second important problem. Invariants aid in hypothesis generation and indexing but give little aid in figure-ground discrimination.

Quasi-invariants were introduced to extend invariants

to more vision problems, e.g. figure-ground discrimination and hypothesis generation in real, complex vision problems. E.g., for five points in a plane there are two invariants. There are no invariants for 3 or 4 points in a plane, although quadrilaterals (4 points in a plane) or triangles (3 points) are important. There are many quasi-invariants; they are widely applicable, e.g. in cases in which there are known to be no invariants. There are 4 quasi-invariants for 4 points in a plane and 2 quasi-invariants for 3 points in a plane.

Quasi-invariants provide a mechanism for figure-ground discrimination with unknown objects with great variation, based on quasi-invariant relations on GC parts. Quasi-invariants provide a mechanism for hypothesis generation by estimating the approximate shape of GC parts in 3-space and shape of objects formed of GC parts. Estimating 3-space shape from image measurements is an inverse process. Indexing is done using partial, structured 3d shape descriptors as in Nevatia and Binford [Nevatia 83]. Structured means part-whole descriptions. Verification, i.e. matching, is accomplished with full 3d shape descriptions. This can be called structured, 3d interpretation by inverse methods.

The structured 3d interpretation paradigm matches object descriptions in 3-space. In contrast, view-sensitive methods match image descriptions. There is much more variation in images than there is in 3-space because of surface markings, pose, lighting, sensor response and obscuration. Our paradigm reduces complexity of matching by reducing complexity of descriptions at the cost of complexity in generating 3d descriptions. For the structured 3d approach to succeed, segmentation, figure-ground discrimination and 3d shape inference must all be effective. Most researchers choose to avoid dependence on effective segmentation, figure-ground discrimination and 3d shape inference. View-sensitive methods suffer complexity of matching complex descriptions by ignoring generating 3d image descriptions. In reality, they cannot avoid grouping, i.e. figure-ground discrimination; it is essential to reduce the combinatorics of combinations of image features. The balance in terms of complexity swings far in favor of generating 3d descriptions. The majority of recognition by aspect graphs uses 3d data with 3d objects, even though it is possible to measure Euclidean invariants directly with 3d data. Recognition from monocular intensity imagery is much more difficult, using 2d data for 3d objects. Structured, 3d interpretation is effective with 3d data, but monocular image intensity data presents a great challenge. Striking results have been achieved with monocular intensity images.

The structured 3d approach reduces computational complexity by reducing the number of combinations of image features by figure-ground discrimination and reduces the number of object hypotheses by hypothesis generation and indexing.

In this paper, we define quasi-invariants, give examples of quasi-invariants, and discuss systematics of the family of quasi-invariants. We also discuss the use of quasi-invariants in computer vision.

2. USE OF INVARIANTS

The use of invariants is discussed to clarify strategies for use of quasi-invariants. Consider an example: the two invariants for 5 points in the plane [Barrett 91].

[Barrett 91] show an example of recognition of aircraft based on the two invariants for 5 points in a plane. They compute invariants for 5 tuples of points in the image and indexing into the database to match invariants for models. An aircraft is not planar but it was possible to find 5 points on a plane.

[Mundy 92] also demonstrate recognition of buildings using invariants for five points in a plane. A rectangular building has only 4 points, but there might be small structures on the roof. An L-shaped building has 6 points in a plane; a U-shaped building has 8 points in a plane. In these examples, there are typically 10,000 points or lines in a large image. The number of combinations of 5 points is $\binom{10,000}{5} \approx 10^{18}$ invariant calculations and indexes. That number is prohibitive. Simple grouping was used, corresponding to curvilinearity and intersection at vertex. Although the paradigm initially was single step indexing, it has been found essential to test hypotheses generated by indexing. For now, points are somewhat difficult to estimate.

3. USE OF QUASI-INVARIANTS

To motivate the mathematical analysis of quasi-invariants, this section examines their use in effective use in computer vision. Quasi-invariants have been used in stereo correspondence [Arnold and Binford 80], in recognition with Bayes nets [Binford 87] and in interpretation of complex objects [Sato and Binford 92a, 92b].

A quasi-invariant was found for two views of an edge in stereo vision [Arnold and Binford 80]. For human stereo, two views of an edge element have approximately the same angle in two views in the canonical stereo coordinate system. This stereo quasi-invariant is nearly invariant. The difference between the two angles has a distribution with 1 degree full width at half maximum. Another stereo quasi-invariant was found for the distance between pairs of edges in two views. See figure 1. The quasi-invariants were incorporated in two systems for stereo reconstruction [Arnold 83, Baker and Binford 81].

Quasi-invariants for angles and ratios of lengths of straight lines were developed. Statistical distributions were incorporated into a Bayes net for recognition of plumbing parts: valve, elbow or other [Binford 87].

Generalized cylinders provide quasi-invariants that group curves on surfaces to form hypotheses of object parts. This provides a viable mechanism for figure-ground discrimination in complex scenes. The generalized cylinder representation generates quasi-invariants for a large class of shapes. Parallel cross sections for cylinders with arbitrary cross sections, SCGCs, straight constant cross section GCs, have edges that are parallel in space. Parallelism is quasi-invariant in projection, as

shown below. Parallel cross sections for SHGCs, straight homogeneous GCs, have edges that are scaled versions of one another. That property is quasi-invariant. It includes the previous case of cylinders. A search is feasible for corresponding pairs of curves that scale. Detecting correspondence appears to have low complexity. Complete 3d descriptions were generated for a number of complex objects [Sato and Binford 92a, 92b].

Statistical distributions for quasi-invariants enable their use in Bayesian networks. It is important to note that successful recognition is not dependent on assumptions behind these statistical distributions, but computational complexity depends on distributions of pose of objects. That is because the statistical distributions are used primarily in indexing where they measure the probability of detection and the probability of false alarms, i.e. the probability that a surface will be viewed from a favorable viewpoint that aids indexing. Indexing operates in the paradigm of hypothesize and test. In typical cases, only part of the information can be used to generate hypotheses with low complexity. There is more information than is used in indexing, adequate information to verify correct identity if the correct hypotheses were known. I.e., typically, if we were told the correct hypothesis, there would be more than enough information to test that hypothesis; verification would require little computation. A key issue is to use all available information in verification, i.e. matching. Quasi-invariants facilitate indexing by building an approximate 3d model from single monocular images or from partial 3d data.

Quasi-invariants exist for a broad class of measurements that parameterize shape, i.e. ratios of dimensions, angles, ratios of curvatures. Quasi-invariants are valuable because they are almost always available.

4. DEFINITION OF INVARIANTS

For intuition in defining quasi-invariants, consider a definition of invariants and a concrete example.

Let A be a collection of appearances (mathematical objects). Let V be a collection of values (mathematical objects). Let $\rho : A \rightarrow V$ be a function that defines an equivalence relation on A :

$$a_1, a_2 \in A; a_2 \approx a_1 \Leftrightarrow \rho(a_2) = \rho(a_1) \quad 1$$

Comment: appearances and values are names for intuition only; they have no mathematical significance.

A mapping $\phi : A \rightarrow V$ is an invariant of the equivalence relation defined by ρ if it is constant on equivalence classes of A . A complete invariant distinguishes any two inequivalent objects.

An invariant satisfies

$$a_1, a_2 \in A; a_1 \approx a_2 \Rightarrow \phi(a_1) = \phi(a_2) \quad 2a$$

A complete invariant satisfies:

$$\phi(a_1) = \phi(a_2) \Leftrightarrow a_1 \approx a_2 \quad 2b.$$

Example

A concrete example will help make the definition clear. For appearances A take the set of all views of all sets of 4 colinear points in 3-space. For the equivalence relation, consider equivalent any two views of 4 colinear points with the cross ratio computed in three space. $\rho \rightarrow R^1$ is the cross ratio computed in three space. Sets of 4 colinear points with different lengths but the same cross ratio are distinct in a Euclidean sense but are equivalent under ρ .

Let G be the set of translations with rigid rotations; it is a noncompact group. Intersect G with the fat sphere, i.e. the portion of the infinite viewing ball beyond a minimum radius. An equivalence class of appearances is a set of 4 colinear points with the same cross ratio computed in three space, with origin in the fat sphere, i.e. a subset of all translations and rotations.

The cross ratio of 4 image points $\phi : A \rightarrow R^1$ is a scalar function that projects 4 colinear points and computes a scalar, i.e. $V = R^1$. The cross ratio of 4 image points is known to be a projective invariant; it is invariant for the equivalence relation defined by ρ , i.e. it is constant on all sets of 4 colinear points with the same cross ratio computed in space.

ϕ , the cross ratio of 4 image points, is complete in this example; ϕ has different values for any pair of 4 colinear points with different values of ρ , the cross ratio computed in three space. The cross ratio is not complete on the other equivalence relation mentioned above, equality of Euclidean lengths of the three intervals of 4 colinear points.

It is useful to think of internal structure independent of external variation. Internal structure is equivalent to the set of equivalence classes determined by ρ , the cross ratio in this example. In other cases, internal structure is parameterized by various measures, e.g. Euclidean lengths of intervals, other shape parameters for more complex structures, or by kinematics or dynamics, e.g. quantum numbers. External variation is determined by G , e.g. the translations with rigid rotations $T_3 \times SO_3$.

The classical definition of invariants is slightly less general. Consider affine invariants, invariant under linear transformations. K is a field of characteristic zero, e.g. the reals R or C , the complex numbers. G is $GL(n, K)$, the general linear group with dimension n over field K . F is a homogeneous form of degree d in n variables $\xi_1 \dots \xi_n$ with coefficients in K . $\sigma \in G$ is a linear transform on an affine space, the action of G on the polynomial ring R , a matrix defined by the rational representation of σ .

g is a relative invariant if $\sigma g = a(\sigma)g$; further, $a(\sigma) = (\det \sigma)^w$. g is an invariant of weight w . g is an absolute invariant if $w = 0$.

Relative invariants on the ring $R[\xi_1 \dots \xi_n]$ are covariants. I.e., covariants involve both coefficients and coordinates. Not every equivalence relation is determined by group action.

5. DEFINITION OF QUASI-INVARIANTS

For intuition, the definition of quasi-invariants (semi-invariants) will be as nearly parallel as possible to the definition of invariants and the example above. Later, the definition of quasi-invariants will be compared to existing mathematical usage of the terms semi-invariants or quasi-invariants.

Let A be collections of appearances (mathematical objects). Let V be collections of values (mathematical objects). Let $\rho : A \rightarrow V$ be a function that defines equivalence classes on A by:

$$a_1, a_2 \in A; a_1 \approx a_2 \Leftrightarrow \rho(a_2) = \rho(a_1).$$

A mapping $\phi : A \rightarrow V$ is a quasi-invariant (semi-invariant) of ρ at $\alpha \in A$ if it is locally constant on equivalence classes of A (defined below) and if ϕ is locally equivalent to ρ at α .

Exposition

ϕ is locally constant, i.e. locally invariant, if the Taylor series for ϕ at $\alpha \in A$ is constant to second order under differential group actions. ϕ is locally equivalent to ρ at α if it has the same Taylor series to first order.

Make definitions to implement those Taylor series. Let $A = R^{m_0}$ and $V = R^{n_0}$. Let $\gamma \in R^{m_0}$ be an infinitely differentiable representation of G with parameterization $u \in R^n$. Let ρ be infinitely differentiable with parameterization $v \in R^n$.

$$\begin{aligned} \phi_i = \phi_i \Big|_{\mu} + \sum_{j=1}^{m_0} \sum_{k=1}^{n_0} \frac{\partial \phi_i}{\partial \gamma_j} \Big|_{\alpha} \frac{\partial \gamma_j}{\partial u_k} \Big|_0 du_k \\ + \sum_{j=1}^{m_0} \sum_{k=1}^{m_0} \sum_{l=1}^{n_0} \sum_{m=1}^{n_0} \frac{\partial^2 \phi_i}{\partial \phi_i \partial \gamma_j} \Big|_{\alpha} \frac{\partial \gamma_j}{\partial u_l} \Big|_0 \frac{\partial \gamma_k}{\partial u_m} \Big|_0 du_l du_m + \dots \end{aligned}$$

A mapping $\phi : A \rightarrow V$ is quasi-invariant if the following two conditions hold:

$$\frac{\partial \phi_i}{\partial u_k} = 0 = \sum_{j=1}^{m_0} \frac{\partial \phi_i}{\partial \gamma_j} \Big|_{\alpha} \frac{\partial \gamma_j}{\partial u_k} \Big|_0 \quad 4a$$

$$\frac{\partial \phi_i}{\partial v_k} = 0 = \sum_{j=1}^{m_0} \frac{\partial \phi_i}{\partial \gamma_j} \Big|_{\alpha} \frac{\partial \gamma_j}{\partial v_k} \Big|_0 \quad 4b$$

Comments

A differentiable representation of G implies a metric and differentiability of $a \in A$. A quasi-invariant is a complete quasi-invariant. It distinguishes any two inequivalent objects locally. A trivial quasi-invariant has the same value on all equivalence classes. Trivial quasi-invariants have no interest.

An example helps to clarify the definition of quasi-invariants. See figure 2a. The collection of appearances A is the set of all views of all sets of three colinear points. G is the set of translations and rigid rotations intersected with the fat sphere. The equivalence classes are defined by ρ : two views are equivalent for 3 colinear points with the same colinear ratio computed in three space. Equivalence classes are all views of 3 colinear

points with the same colinear ratio computed in three space.

$\phi : A \rightarrow R^1$ is a scalar function, e.g. the projected colinear ratio of 3 projected image points. ϕ , the projected colinear ratio is quasi-invariant at $z = \infty, \theta = 0$; the gradient is zero there, both partials vanish. The projected colinear ratio is constant to second order at $z = \infty, \theta = 0$ over equivalence classes, colinear triples with the same colinear ratios evaluated in three space. The projected colinear ratio is even constant to third order at $z = \infty, \theta = 0$ i.e. second partials are zero also. The projected colinear ratio at $z = \infty, \theta = 0$ is linearly equivalent to the true colinear ratio evaluated in three space; it has the same value and first derivative at $z = \infty, \theta = 0$. The colinear ratio is locally complete on A . It takes a different value near $z = \infty, \theta = 0$ for non-equivalent triples of points, views of 3 colinear points with different ratio. The colinear ratio is not complete on equivalence classes defined by Euclidean length of intervals. Two sets of 3 colinear points with the same colinear ratio could differ in Euclidean lengths of intervals.

Previous Definitions of Semi-Invariants

Two uses exist for the term semi-invariant. The definition similar to this definition is semi-invariants of Lie groups. Let G be a Lie Group. Let K be R or C . ρ is a differentiable representation of G , $\rho(G) \subset GL(n, K)$, the general linear group. ρ defines an action of G on $K[x_1, \dots, x_n]$

For infinitesimal X_a , a semi-invariant or invariant element f satisfies: $X_a f = 0$ ($\forall X_a$) or: $X_a f = \alpha_k f, \alpha \in K$ ($\forall X_a$)

Another definition is equivalent to relative invariant; that definition is uninteresting. R is a commutative ring. The group action $\sigma \in G$ defines an automorphism: $f \rightarrow \sigma f \in R$ and $\sigma(\tau f) = (\sigma\tau)f$ for $\sigma, \tau \in G$.

f in R is G -semi-invariant if for each $\sigma \in G$, f is invariant up to an invariant multiplier depending on σ : $\sigma f = a(\sigma)f$. That definition is equivalent to relative invariant. There is an integer w such that $a(\sigma) = (\det(\sigma))^w$.

Stability of Quasi-Invariants

The intent of quasi-invariants is that a quasi-invariant ϕ approximates a 3-space measurement ρ . For ρ to be interesting, it must be a Euclidean invariant. E.g. as above, choose ρ the colinear ratio measured in 3-space, a Euclidean invariant. The image colinear ratio ϕ approximates ρ .

As defined, quasi-invariants are local invariants. Continuity is a very weak condition. Even if ϕ has continuous derivatives of all orders, it could be very badly behaved, i.e. ϕ could be a poor approximation to ρ except very near the quasi-invariant. The utility of quasi-invariants depends on their stability. It turns out that many quasi-invariants are quite stable.

For quasi-invariants to be useful, the quasi-invariant

ϕ must be a reasonable approximation to ρ over a large part of the parameter range. In equation ???, the second term is zero because the gradient is zero. The range over which the quasi-invariant is useful is determined by the range over which the quadratic term and all higher terms have a sum small compared to the constant term, e.g. 1/3 of the constant term. For quasi-invariant examples of interest, it will be useful to examine the quadratic term to determine the range of stability. By definition, the linear term is zero.

That condition is related to a probabilistic interpretation of quasi-invariants. In many situations, it is not possible to restrict very much the viewing conditions of the observer relative to the object. If it were possible to restrict the view, that would be very useful and could probably be included in an analysis like this. For a quasi-invariant to be useful, it should be a good approximation over a large fraction of views, i.e. over a large part of the fat view sphere, the hollow viewing ball.

For several of the quasi-invariants considered here, all second partial derivatives vanish. This makes the order of zero higher at m , hence intuitively stronger. The analysis of the range of useful quasi-invariance then requires third-order differentiability and equivalent behavior for third-order terms of the Taylor expansion. There is an equivalent definition for third-order differentials. Even analytic functions contain many wildly misbehaving functions. These conditions on differentials are much stronger than continuity. Fortunately, those differentials are small enough for quasi-invariants to be quite useful.

6. SUMMARY INVARIANTS AND QUASI-INVARIANTS

For an excellent summary of results in invariants and their relations to computer vision, see [Mundy 91a], especially the introduction [Mundy and Zisserman 91b].

Colinear Points

For the line, there are no invariants for 1, 2 or 3 colinear points. For 4 colinear points, the cross ratio is invariant. For more than 4 points, other invariants exist.

For 3 points, the colinear ratio is quasi-invariant. For more than 3 points, quasi-invariants and invariants exist.

Coplanar Points

For coplanar point sets, there are invariants for lines embedded in the plane. Colinearity of three or more points is invariant. The invariants of the line are inherited in the plane.

For coplanar point sets that are not colinear, there are no invariants for 1, 2, 3, 4 points. There are 2 invariants for 5 points. Because lines and points are dual, there are also 2 invariants for 5 non-degenerate lines. For more than 5 points or more than 5 lines, more invariants exist. The invariants assume correspondence of points.

With one fewer point, quasi-invariants exist. For 4

points in the plane, 4 quasi-invariants exist. 4 non-degenerate lines form a quadrilateral, hence 4 quasi-invariants exist for four lines in the plane.

With 3 points, the angle and ratio of lengths or longitudinal and transverse ratios are two quasi-invariants. 3 points define the simplex, i.e. quasi-invariants exist on any face of a polyhedron. 3 non-degenerate lines form a triangle; for 3 non-degenerate lines there are two quasi-invariants. Just as with invariants, quasi-invariants relate corresponding points.

For 5 or more points, there are invariants and quasi-invariants.

Plane Curves

Because all conics are equivalent under projection, there are no invariants for conics. For two conics there is an invariant involving the bi-tangents, lines tangent to two conics. There is an invariant for a conic and two straight lines.

One quasi-invariant exists for a single conic. E.g. for an ellipse, the ratio of minor/major axis is quasi-invariant.

Invariants exist for algebraic curves of third and higher order. Classical results of algebraic curves give an extensive theory of invariants.

3 space: Single View

Invariants exist for algebraic surfaces. They appear to be sensitive to measurement error. Invariants are known not to exist for single views of general point sets in 3D.

A large number of quasi-invariants are known for 3d surfaces. The utility of quasi-invariants is related to the generalized cylinder representation of object parts. [Sato and Binford 92a]

3 space: Multiple Views

While no invariants exist for single views of general point sets in 3D, there are useful invariants for multiple views [Barrett 91].

Quasi-invariants for stereo were found for angles of edges and intervals between edges, as mentioned above [Arnold and Binford 80].

7. QUASI-INVARIANTS FOR COPLANAR POINTS

For three points, there are two quasi-invariants. For four coplanar points, there are four quasi-invariants. Figure 2b shows four coplanar points in a general position relative to the viewframe: $\vec{p}_0, \vec{p}_1, \vec{p}_2, \vec{p}_3$. Those four points in a body-centered frame are: $\vec{p}_0^b, \vec{p}_1^b, \vec{p}_2^b, \vec{p}_3^b$.

$$\begin{aligned} \vec{p}_0^b &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}; \quad \vec{p}_1^b = \begin{bmatrix} l_1 \\ 0 \\ 0 \end{bmatrix}; \quad \vec{p}_2^b = \begin{bmatrix} l_2 \cos \alpha \\ l_2 \sin \alpha \\ 0 \end{bmatrix}; \\ \vec{p}_3^b &= \begin{bmatrix} l_3 \cos \beta \\ l_3 \sin \beta \\ 0 \end{bmatrix} \end{aligned} \quad 7.1.$$

$$\begin{aligned} \vec{q}_1 &\equiv \vec{p}_1 - \vec{p}_0; \quad \vec{q}_2 \equiv \vec{p}_2 - \vec{p}_0; \quad \vec{q}_3 \equiv \vec{p}_3 - \vec{p}_0; \\ l_1 &\equiv |\vec{q}_1|; \quad l_2 \equiv |\vec{q}_2|; \quad l_3 \equiv |\vec{q}_3|; \\ \hat{e}_1 &\equiv \frac{\vec{q}_1}{l_1}; \quad \hat{e}_3 \equiv \frac{\vec{q}_1 \times \vec{q}_2}{l_1 l_2}; \quad \hat{e}_2 \equiv \hat{e}_3 \times \hat{e}_1. \\ \cos \alpha &\equiv \frac{\vec{q}_2 \cdot \vec{q}_1}{l_1 l_2}; \quad \cos \beta \equiv \frac{\vec{q}_3 \cdot \vec{q}_1}{l_1 l_3}; \\ \hat{n} &\equiv \frac{\vec{q}_1 \times \vec{q}_2}{|\vec{q}_1| |\vec{q}_2|}. \end{aligned}$$

Choose the view frame as in figure 2c with an orthonormal triple $\hat{E}_1, \hat{E}_2, \hat{E}_3$, with \hat{E}_1 along an arbitrary axis ("right") in the image plane, and with \hat{E}_3 along the principal axis, the normal to the plane of projection, toward the center of projection. Then \hat{E}_2 is "down" in the image plane in a right-handed frame.

Consider four coplanar points initially in a frame aligned with the view frame;

$$\hat{e}_1^b = \hat{E}_1; \quad \hat{e}_2^b = \hat{E}_2; \quad \hat{e}_3^b = \hat{E}_3.$$

Transform the coplanar points into the four points in a general frame:

$$\vec{p}_0^b \rightarrow \vec{p}_0; \quad \vec{p}_1^b \rightarrow \vec{p}_1; \quad \vec{p}_2^b \rightarrow \vec{p}_2; \quad \vec{p}_3^b \rightarrow \vec{p}_3.$$

Rotate through Euler angles, ϕ, θ, ψ and translate to \vec{p}_0 . Rotate first about \hat{e}_3^b through ϕ to align $\hat{e}_2^b \rightarrow \hat{e}_2^b$ perpendicular to the plane of \hat{e}_3^b and \hat{n} . Then rotate by θ about the transformed \hat{e}_2^b , i.e. about \hat{e}_2^b to bring the normal of the view frame into the normal to the plane of the four coplanar points: $\hat{e}_3^b \rightarrow \hat{n}$. Rotate by ψ about \hat{n} .

The rotation matrices are R_ϕ, R_θ, R_ψ :

$$\begin{aligned} R_\phi &= \begin{bmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix}; \\ R_\theta &= \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix}; \\ R_\psi &= \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad 7.2.$$

The combined rotation matrix is

$$R = R_\psi R_\theta R_\phi \quad 7.3.$$

The coordinates of the points $\vec{p}_0, \vec{p}_1, \vec{p}_2, \vec{p}_3$ in the view frame are:

$$\vec{p}_i = R_\psi R_\theta R_\phi \vec{p}_i^b + \vec{p}_0 \quad 7.4.$$

Projected coordinates of the points are:

$$\bar{p}_i = \frac{Z_0}{z_i} \vec{p}_i; \quad 7.5$$

where Z_0 is the image distance, z_i is the distance to point \vec{p}_i in the view frame. Define image difference vectors:

$$\bar{Q}_1 \equiv \bar{p}_1 - \bar{p}_0; \quad \bar{Q}_2 \equiv \bar{p}_2 - \bar{p}_0; \quad \bar{Q}_3 \equiv \bar{p}_3 - \bar{p}_0. \quad 7.6$$

Quasi-Invariants for Four Coplanar Points

Define two quasi-invariants μ_1, μ_2 for images of four coplanar points in a general frame, in terms of image difference vectors $\bar{Q}_1, \bar{Q}_2, \bar{Q}_3$:

$$\begin{aligned}\mu_1 &= \frac{-Q_{22}Q_{31} + Q_{21}Q_{32}}{-Q_{22}Q_{11} + Q_{21}Q_{12}}, \\ \mu_2 &= \frac{-Q_{32}Q_{11} + Q_{12}Q_{31}}{-Q_{22}Q_{11} + Q_{21}Q_{12}},\end{aligned}\quad 7.7$$

where Q_{ij} is the j th component of the i th image difference vector \bar{Q}_i . There are two additional quasi-invariants for three coplanar points. For now, consider μ_1, μ_2 . They are strong quasi-invariants.

To show that μ_1, μ_2 are quasi-invariant under projection at $z = \infty, \theta = 0$, show that they are equal to the Euclidean invariants and that the gradient with respect to view frame parameters vanishes:

$$\begin{aligned}\mu_1 &\rightarrow \mu_1^b; \mu_2 \rightarrow \mu_2^b; z \rightarrow \infty, \theta \rightarrow 0, \\ \frac{\partial \mu_i}{\partial x} = 0 &= \frac{\partial \mu_i}{\partial y} = \frac{\partial \mu_i}{\partial z} = \frac{\partial \mu_i}{\partial \phi} = \frac{\partial \mu_i}{\partial \theta} = \frac{\partial \mu_i}{\partial \psi}; \\ z &\rightarrow \infty; \theta \rightarrow 0.\end{aligned}\quad 7.8$$

Substitute variables from equations 7.6, 7.5, 7.4, 7.2 and 7.1 into equation 7.7. Conditions 7.8 were demonstrated in map; however, only the condition $z \rightarrow \infty$ is necessary; the condition $\theta \rightarrow 0$ is not necessary.

A stronger result was proved: all second derivatives vanish at $z \rightarrow \infty$.

$$\frac{\partial^2 \mu_i}{\partial \mu_i \partial v_j} = 0; \quad v_j \in \{x, y, z, \phi, \theta, \psi\}.$$

These are strong quasi-invariants, constant to third order.

Three Coplanar Points

With three coplanar points with the notation above, there are two independent quasi-invariants at $z \rightarrow \infty, \theta \rightarrow 0$, the angle between the two difference vectors and ratio of lengths of difference vectors:

$$\cos \gamma = \frac{\bar{Q}_2 \cdot \bar{Q}_1}{|\bar{Q}_2||\bar{Q}_1|}; \quad \rho^2 = \frac{\bar{Q}_2 \cdot \bar{Q}_2}{\bar{Q}_1 \cdot \bar{Q}_1}.$$

Those quasi-invariants are equivalent to the ratios of longitudinal and transverse components of \bar{Q}_2 to the length of \bar{Q}_1 .

$$\rho_a = \frac{\bar{Q}_2 \cdot \bar{Q}_1}{|\bar{Q}_1|^2}; \quad \rho_b = \frac{\bar{Q}_2 \times \bar{Q}_1}{|\bar{Q}_1|^2}.$$

Both γ, ρ satisfy the condition that the image estimates are identical to the Euclidean invariant at $z \rightarrow \infty, \theta \rightarrow 0$.

$$\begin{aligned}\gamma &\rightarrow \gamma^b; \rho \rightarrow \rho^b; z \rightarrow \infty, \theta \rightarrow 0, \\ \frac{\partial \gamma}{\partial x} = 0 &= \frac{\partial \gamma}{\partial y} = \frac{\partial \gamma}{\partial z} = \frac{\partial \gamma}{\partial \phi} = \frac{\partial \gamma}{\partial \theta} = \frac{\partial \gamma}{\partial \psi}; \\ z &\rightarrow \infty; \theta \rightarrow 0.\end{aligned}$$

$$\frac{\partial \rho}{\partial x} = 0 = \frac{\partial \rho}{\partial y} = \frac{\partial \rho}{\partial z} = \frac{\partial \rho}{\partial \phi} = \frac{\partial \rho}{\partial \theta} = \frac{\partial \rho}{\partial \psi};$$

$$z \rightarrow \infty; \theta \rightarrow 0.$$

$$\rho_a \rightarrow \rho_a^b; \rho_b \rightarrow \rho_b^b; z \rightarrow \infty, \theta \rightarrow 0,$$

$$\begin{aligned}\frac{\partial \rho_i}{\partial x} = 0 &= \frac{\partial \rho_i}{\partial y} = \frac{\partial \rho_i}{\partial z} = \frac{\partial \rho_i}{\partial \phi} = \frac{\partial \rho_i}{\partial \theta} = \frac{\partial \rho_i}{\partial \psi}; \\ z &\rightarrow \infty; \theta \rightarrow 0.\end{aligned}\quad 7.9$$

Figure 3 shows the two-dimensional distribution of γ, ρ in terms of fractions of the view sphere. The two quasi-invariants are not strong quasi-invariants, i.e. second derivatives are non-vanishing. They are nearly independent.

The Image Colinear Ratio is Quasi-Invariant

The cross ratio is invariant for four colinear points. The image colinear ratio is quasi-invariant under projection for three colinear points at $z_2 = \infty, \theta = 0$ on the fat sphere.

Three points in space x_0, x_1, x_2 project to three image points X_0, X_1, X_2 .

$$\begin{aligned}\bar{p}_0 &= \begin{bmatrix} x_0 \\ 0 \\ z_0 \end{bmatrix} \rightarrow \bar{P}_0 = \begin{bmatrix} X_0 \\ 0 \\ Z \end{bmatrix} = \begin{bmatrix} \frac{z}{z_0} x_0 \\ 0 \\ Z \end{bmatrix} \\ \bar{p}_1 &= \begin{bmatrix} x_1 \\ 0 \\ z_1 \end{bmatrix} \rightarrow \bar{P}_1 = \begin{bmatrix} X_1 \\ 0 \\ Z \end{bmatrix} = \begin{bmatrix} \frac{z}{z_1} x_1 \\ 0 \\ Z \end{bmatrix} \\ \bar{p}_2 &= \begin{bmatrix} x_2 \\ 0 \\ z_2 \end{bmatrix} \rightarrow \bar{P}_2 = \begin{bmatrix} X_2 \\ 0 \\ Z \end{bmatrix} = \begin{bmatrix} \frac{z}{z_2} x_2 \\ 0 \\ Z \end{bmatrix}\end{aligned}\quad 7.10$$

Represent the line in space by the center point x_2 , two line segments l_1, l_2 , and two angles, ϕ , azimuth and θ , inclination. line segment: $x_2, l_1, l_2, \phi, \theta$ Choose \hat{e}_1 and x_1 along the line, and \hat{e}_2 and x_2 normal to the line. Thus $\phi = 0$.

$$x_1 = x_2 - l_1 \cos \theta; \quad z_1 = z_2 - l_1 \sin \theta \quad 7.11$$

$$x_3 = x_2 + l_2 \cos \theta; \quad z_3 = z_2 + l_2 \sin \theta$$

The colinear ratio is computed in 3-space:

$$r = \frac{l_2}{l_1 + l_2}$$

The image colinear ratio is computed in the image by the ratio of one interval in the image to the sum of intervals.

$$R = \frac{(x_3 z_2 - x_2 z_3) z_3 z_1}{(x_3 z_1 - x_1 z_3) z_3 z_2} \quad 7.12$$

$$R = \frac{X_3 - X_2}{X_3 - X_1} = \frac{\frac{z x_3}{z_3} - \frac{z x_2}{z_2}}{\frac{z x_3}{z_3} - \frac{z x_1}{z_1}}$$

From equation 7.12, calculate the ratio using equation 7.11 for x_1, x_3 :

$$R = \frac{l_2}{l_1 + l_2} \frac{z_2 \cos \theta - x_2 \sin \theta - l_1 \sin \theta (\cos \theta - \frac{x_2}{z_2} \sin \theta)}{z_2 \cos \theta - x_2 \sin \theta} \quad 7.13$$

Now, determine that the colinear ratio is quasi-invariant by computing its derivatives at $z_2 = \infty, \theta = 0$.

$$\frac{dR}{dz_2}, \frac{dR}{d\theta} \text{ at } z_2 = \infty; \theta = 0.$$

The colinear ratio is quasi-invariant at $z_2 = \infty; \theta = 0$ if derivatives vanish. It is an affine invariant.

Now consider the worst case, i.e. the largest values of l_1/z_2 . Human and aerial limits that follow are extremes. For most objects, l_1/z_2 values are much smaller.

- For human perception, a line 20 cm at 30 cm $l_1/z_2 \leq .1$ subtends a large visual angle.
- For aerial photography with a 9" by 9" image with a focal length of $f=12"$, $l_1/z_2 \leq 4.5/12 = .375$. At 6 miles altitude, $l_1 + l_2 \geq 4.5$ miles.

Consider ϵ contours of the colinear ratio:

$$R = \frac{l_2}{l_1 + l_2} (1 - \epsilon).$$

Again, consider $x_2 = 0$ for simplicity.

$$\frac{l_1}{z_2} \sin \theta = \epsilon; \quad \sin \theta = \frac{\epsilon}{\frac{l_1}{z_2}}.$$

- The colinear ratio is invariant to 10% over the human limit $l_1/z_2 \leq .1$.
- The colinear ratio is invariant to 30% over almost the full range of interest for aerial photography.

The colinear ratio for 3 colinear points is a strong quasi-invariant. Figure 4 shows ϵ contours for .1, .2, .3. The area above and to the left of a contour is the range of angles and distances on the fat sphere for which the colinear ratio is less than ϵ .

Variance of Colinear Ratio

Now compute the variance of the colinear ratio from the true value $l_2/(l_1 + l_2)$:

$$R = \frac{l_2}{l_1 + l_2} (1 - \frac{l_1}{z_2} \sin \theta)$$

$$V(R) = \int_0^q d\left(\frac{l_1}{z_2}\right) \int_0^{\frac{\pi}{2}} d\theta \left(\frac{l_1}{z_2}\right)^2 \sin^2 \theta.$$

Assume that the distribution of viewpoints is uniform on $\frac{l_1}{z_2}$ over $[0, q]$ where q is .1 for human vision; q is .375 for aerial photography. Assume that the distribution is uniform on θ

$$V(R) = \int_0^q d\left(\frac{l_1}{z_2}\right) \left(\frac{l_1}{z_2}\right)^2 \int_0^{\frac{\pi}{2}} d\theta \sin^2 \theta$$

$$= \frac{\pi}{4} \int_0^q d\left(\frac{l_1}{z_2}\right) \left(\frac{l_1}{z_2}\right)^2.$$

$$V = \frac{\pi}{12} \left(\frac{l_1}{z_2}\right)^3 \Big|_0^q$$

- For the human limit, $q = .1$

$$V = \frac{\pi}{12} .1^3 = .000262.$$

The standard deviation is $\sigma = .016$; invariant to 1.6%.

- For aerial photography, $q = .375$

$$V = \frac{\pi}{12} .375^3 = .0517.$$

The standard deviation is $\sigma = .117$, invariant to 12%.

These values are very tight, nearly invariant.

Length is not Quasi-Invariant

Projected length is not a quasi-invariant under perspective or weak perspective. Projected length does equal Euclidean length at one distance but the gradient with respect to view parameters does not vanish there. At $z = \infty$, the gradient does vanish, but the value of projected length does not converge to Euclidean length.

References

- [Arnold 83] R.D.Arnold; "Automated Stereo Perception"; PhD Dissertation, Stanford University, 1983.
- [Arnold and Binford 80] R.D.Arnold and T.O.Binford; "Geometric Constraints in Stereo Vision"; Proc SPIE v 238, 281, 1980.
- [Baker and Binford 81] H.Baker and T.O.Binford; H. Baker and T.O. Binford, "Depth from Edge and Intensity Based Stereo," 7th Int. Joint Conf. on Artificial Intelligence, Vancouver, Canada, August 24-28, 1981, pp 631-636.
- [Barrett 91] E.B.Barrett, P.M.Payton, N.N.Haag, M.H.Brill; "General Methods for Determining Projective Invariants in Imagery"; CVGIP: Image Understanding, v 53, pp 46-65, 1991.
- [Binford 87] T.O.Binford, T.S.Levitt, W.B.Mann; "Bayesian Inference in Model-Based Machine Vision"; Uncertainty in Artificial Intelligence, 3, pp 73-95, 1987.
- [Mundy 91a] J.L.Mundy and A. Zisserman; "Geometric Invariance in Computer Vision"; MIT Press, 1991.
- [Mundy and Zisserman 91b] J.L.Mundy and A. Zisserman; "Introduction - Towards a New Framework for Vision"; *Geometric Invariance in Computer Vision*; pp 1-39 MIT Press, 1991.
- [Mundy 92] J.L.Mundy, P.M.Payton, M.H.Brill, E.A.Barret, R.P.Welty; "3-D Model Alignment without Computing Pose"; Proc DARPA IU Workshop, pp 727-735, 1992.
- [Sato and Binford 92a] H. Sato and T.O.Binford; "On finding the ends of SHGCs in an Edge Image"; Proc DARPA IU Workshop, pp 379-390, 1992.
- [Sato and Binford 92b] H.Sato and T.O.Binford; "Builder-I: A system for the extraction of SHGC objects in an edge image"; Proc DARPA IU Workshop, pp 779-792, 1992.

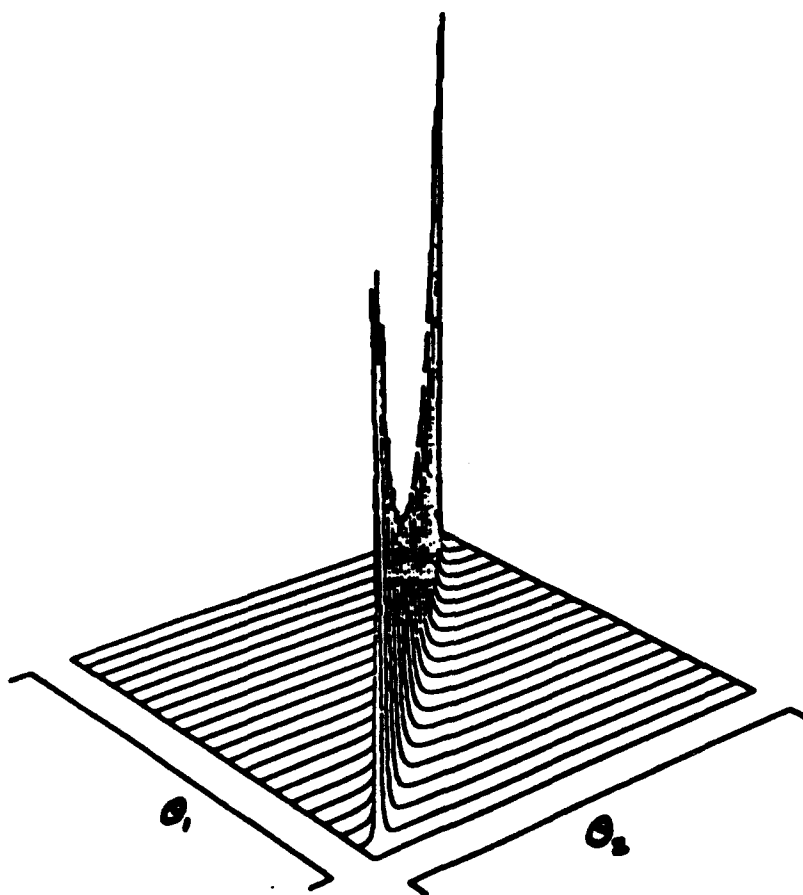


Figure 1: Stereo quasi-invariant: difference in angles of two views of one vector.



Figure 2a: Three colinear points.

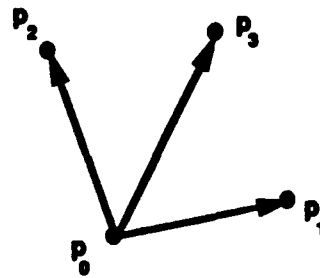


Figure 2b: Four coplanar points.

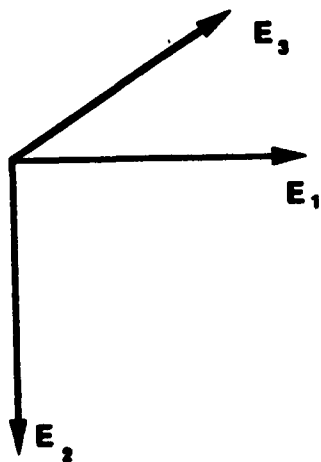
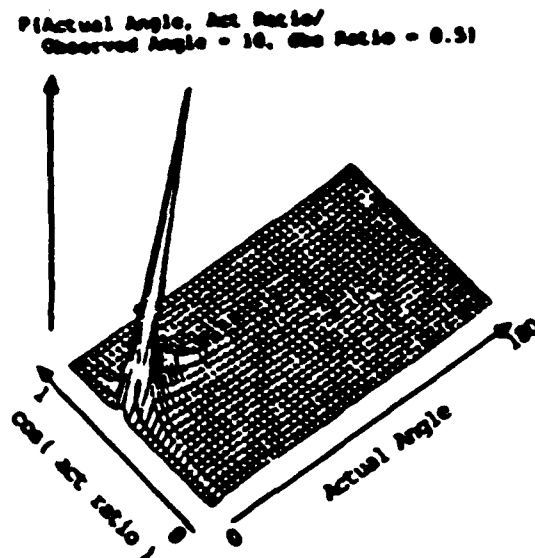


Figure 2c: Egocentric coordinate frame.



Joint probability $p(\text{observed angle and ratio} | \text{actual angle and ratio})$

Figure 3: 2-d distribution of angle and length ratio for three points.

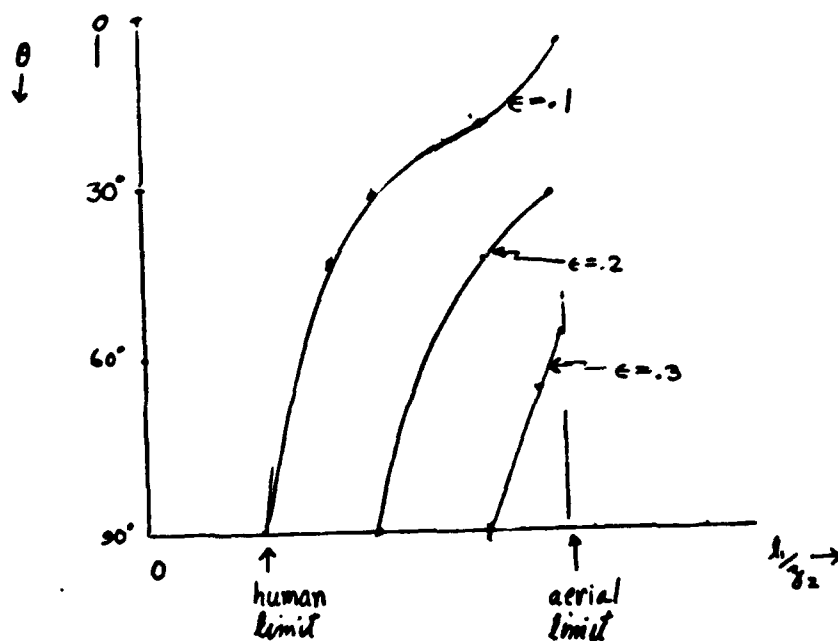


Figure 4: Epsilon contours of colinear ratio quasi-invariant.
All values above and left of epsilon contour lie within the contour.

A Spherical Representation for the Recognition of Curved Objects

H. Delingette, M. Hebert, K. Ikeuchi

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA15213

Abstract¹

In this paper, we introduce a new surface representation for recognizing curved objects. Our approach begins by representing an object by a discrete mesh of points built from range data or from a geometric model of the object. The mesh is computed from the data by deforming a standard shaped mesh, for example, an ellipsoid, until it fits the surface of the object. We define local regularity constraints that the mesh must satisfy. We then define a canonical mapping between the mesh describing the object and a standard spherical mesh. A surface curvature index that is pose-invariant is stored at every node of the mesh. We use this object representation for recognition by comparing the spherical model of a reference object with the model extracted from a new observed scene. We show how the similarity between reference model and observed data can be evaluated and we show how the pose of the reference object in the observed scene can be easily computed using this representation.

1.0 Introduction

In this paper, we propose a new representation for 3-D objects which is suitable for object recognition and pose determination. Many representations have been proposed to address this recognition problem. Local approaches attempt to represent objects as sets of primitives such as faces or edges. Most early local methods handle polyhedral objects and report effective and encouraging results. Representative systems include [12][19][15]. Few systems can handle curved surfaces. Examples include early work in which primitive surfaces enclosed by orientation discontinuity boundaries are extracted from range data [20]. Other systems determine primitive surfaces which satisfy planar or quadric equations [9]. Techniques based on differential geometry such as [3] segment range images using Gaussian curvatures.

The global methods assume one particular coordinate system attached to an object and represent the object as an implicit or parametric function in this coordinate sys-

tem. The resulting representation is global in that the implicit function represents the entire shape of the object or of a large portion of the object. The generalized cylinder (GC) is a representative of this group. Although encouraging results have been obtained in recognizing GCs in intensity images, using generalized cylinders for recognition is difficult due to the difficulty of extracting GC parameters from input images.

Superquadrics (SQ) representation also belongs to the class of global representations [21]. The SQs represent a limited set of shapes which can be extended by adding parameters to the generic implicit equation of SQs. A possible extension is to segment objects into sets of superquadrics [10], although the computational complexity of the scene analysis may become prohibitive. An interesting attempt to handle a large class of natural objects is discussed in [4].

EGI and CEGI map surface orientation distributions to the Gaussian sphere [13] [14] [17] [16]. Since the Gauss map is independent of translation, the representation is quite suitable to handle convex curved objects. In this case, recognition proceeds by finding the rotation that maximizes the correlation between two EGIs [14][7]. However, when part of the object is occluded, these techniques cannot reliably extract the representation.

Recently, new approaches based on the idea of fitting a bounded algebraic surface of fixed degree to a set of data points [22][23][11] have been proposed. Although encouraging results have been obtained in this area, more research is needed in the areas of bounding constraints, convergence of surface fitting, and recognition before this approach becomes practical. For a survey of other techniques that can be used for global surface fitting, see [5].

Another class of approaches attempts to match sets of points directly without any prior surface fitting. An example is the work by Besl [2] in which the distance between point sets is computed and minimized to find the best transformation between model and scene.

Our approach is a combination of the point set matching and of the original EGI approach. As in the case of the point set matching, we want to avoid fitting analytical surfaces to represent an object. Instead, we use a representation that simply consists of a collection of points, or nodes, arranged in a mesh covering the entire surface of the object. This has the advantage that the object can have any arbitrary shape, as long as that shape is topologically equivalent to the sphere. To avoid problems with variable density of nodes on the mesh, we need

¹This research was supported in part by the DARPA Image Understanding Program, through ARPA Order No. 4976, and monitored by the Air Force Avionics Laboratory under contract F33615-87-C-1499, and in part by DARPA under contract DACA 76-89-C-0014 monitored by the Army Topographic Engineering Center.

The views and conclusions contained in this report are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA or the U.S. Government.

to define regularity constraints that must be enforced. We use an extension of the deformable surfaces algorithms introduced in [8] to compute the meshes. As in the EGI algorithms, each node of the mesh is mapped onto a regular mesh on the unit sphere, and a measure of local surface curvature, the *simplex angle*, is stored at the corresponding node on the sphere. We call the corresponding spherical representation the *Simplex Angle Image* (SAI). Finally, we define the regularity constraints such that if \mathcal{M} is the mesh representing an object, and \mathcal{M}' is the mesh representing the same object after transformation by a combination of rotation, translation, and scaling, then the corresponding distributions of simplex angles on the spherical representations S and S' are the same up to a rotation. Therefore, to determine whether two objects are the same, we need only compare the corresponding spherical distributions.

This approach appears similar to the EGI approach. However, one fundamental difference is that a unique mesh, up to rotation, translation and scale, can be reconstructed from a given SAI. In the case of the EGI, this property is true only for convex objects. Another fundamental difference is that the SAI preserves connectivity in that patches that are connected on the surface of the input object are still connected in the spherical representation. The latter is the main reason why our approach can be applied to arbitrary non-convex objects. Connectivity conservation is also the reason why the SAI can be used for recognition even in the presence of significant occlusion, as we will see later in the paper, whereas EGI and other global representations cannot.

The paper is organized as follows. In Section 2.0, we describe a simple representation of closed 2-D curves which we extend to three dimensional surfaces in Section 3.0. In Sections 3.0 to 5.0 we describe the fundamentals of the SAI algorithms in the case of complete object models. In Section 4.0, we show how to obtain SAIs from range data. In Section 5.0, we describe the SAI matching. We address the problem of occlusion and partial models in Section 6.0.

2.0 Representation of 2-D Curves

A standard approach to representing and recognizing contours is to approximate contours by polygons, and to compute a quantity that is related to the curvature of the underlying curve. The similarity between contours can then be evaluated by comparing the distribution of curvature measurement at the vertices of the polygons. In this section, we introduce the basic concepts that can be used to manipulate polygonal representations of contours. The concepts discussed in this section are well known and have been studied extensively. Our purpose here is to introduce them in a way that facilitates their extension to three-dimensional surfaces.

In order to quantify the curvature of a contour, we use the angle ϕ between consecutive segments. Like the curvature, the angle ϕ is independent of rotation and translation. One problem is that if the lengths of the segments representing the curve are allowed to vary, the value of ϕ depends not only on the shape of the curve but also on the distribution of points on the curve. One way to enforce uniform distribution is to impose a *local regularity* condition on the distribution of vertices. The local regularity condition simply states that all the segments must have the same length. Another geometric definition of this

condition is illustrated in Figure 1. The condition that the length of the two segments PP_1 and PP_2 are the same is equivalent to the condition that the projection of node P on the line joining its two neighbors P_1 and P_2 coincides with the center of P_1 and P_2 . This is obviously a more complicated way of formulating the simple regularity condition, but it will become useful when we extend this notion to three dimensions.

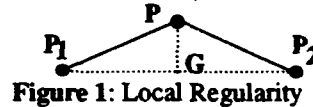


Figure 1: Local Regularity

The last step in representing two-dimensional contours is to build a circular representation that can be used for recognizing contours. Let us assume that the contour is divided into N segments with vertices P_1, \dots, P_N , and with corresponding angles ϕ_1, \dots, ϕ_N . Let us divide the unit circle using N equally spaced vertices C_1, \dots, C_N . Finally, let us store the angle ϕ_i associated with P_i at the corresponding circle point C_i (Figure 2). The circular representation of the contour is invariant by rotation, translation, and scaling. As the density of points increases, the circular representations of two contours are identical up to a rotation of the unit circle. This property allows for comparing contours by declaring that two contours are identical if there exists a rotation of the unit circle that brings their representation in correspondence. Also, when comparing contours, the distribution of the vertices C_i on the circle must be uniform. We refer to this property as *global regularity*.

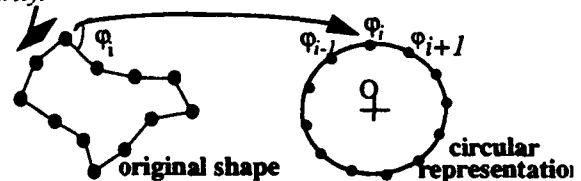


Figure 2: Mapping from Shape and to Unit Circle

3.0 Representation of 3-D Surfaces

In this section we extend to three-dimensional surfaces the concepts of curvature measure (Section 3.3), local and global regularity (Section 3.2 and Section 3.1), and circular representation (Section 3.4). We consider the case of representing surfaces topologically equivalent to the sphere. (Cases in which only part of the surface is visible will be addressed in Section 6.0.) Detailed presentations of the basic results on semi-regular tessellations, triangulations, and duality can be found in [18][24][25].

3.1 Global Regularity and Mesh Topology

A natural discrete representation of a surface is a graph of points, or mesh, such that each node is connected to each of its closest by an arc of the graph. It is desirable for many algorithms to have a constant number of neighbors at each node. We use a class of meshes such that each node has exactly three neighbors. Such meshes can always be constructed as the dual of a triangulation.

As mentioned in the previous section, global regularity can be easily achieved in two dimensions since a curve can always be divided into an arbitrary number of segments of equal length. It is well known that only approximate global regularity can be achieved in three dimensions. We use a quasi-regular triangulation con-

structed by subdividing each triangular face of a 20-face icosahedron into N^2 smaller triangles. The final mesh is built by taking the dual of the $20N^2$ faces triangulation, yielding a mesh with the same number of nodes. For the experiments presented in this paper, we used a subdivision frequency of $N = 7$ for a total number of nodes of 980.

3.2 Local Regularity

The definition of local regularity in three dimensions is a straightforward extension of the definition of Section 2.0. Let P be a node of the mesh, P_1, P_2, P_3 be its three neighbors, G be the centroid of the three points, and Q be the projection of P on the plane defined by P_1, P_2 , and P_3 (Figure 3). The local regularity condition simply states that Q coincides with G . This is the same condition as in two dimensions, replacing the triangle (P_1, P_2, P) of Figure 1 by the tetrahedron (P_1, P_2, P_3, P) . The local regularity condition is invariant by rotation, translation, and scaling.

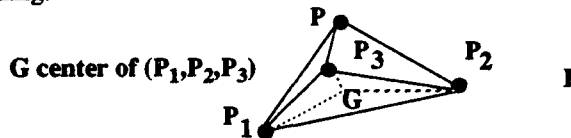


Figure 3: Local Regularity in Three Dimensions

3.3 Discrete Curvature Measure

Before defining the discrete curvature measure, we need to define some notations (Figure 4 (a)). Let P be a node of the mesh, P_1, P_2, P_3 its three neighbors, O the center of the sphere circumscribed to the tetrahedron (P, P_1, P_2, P_3) , Z the line passing through O and through the center of the circle circumscribed to (P_1, P_2, P_3) . Now, let us consider the cross section of the surface by the plane Π containing Z and P . The intersection of Π with the tetrahedron is a triangle. One vertex of the triangle is P , and the base opposite P is in the plane (P_1, P_2, P_3) (Figure 4 (b)). We define the angle ϕ_0 as the angle between the two edges of the triangle intersecting at P . By definition, ϕ_0 is the discrete curvature measure at node P . We call ϕ_0 the simplex angle at P , since it is the extension to a three-dimensional simplex, the tetrahedron, of the notion introduced in Figure 1 for a two-dimensional simplex, the triangle.

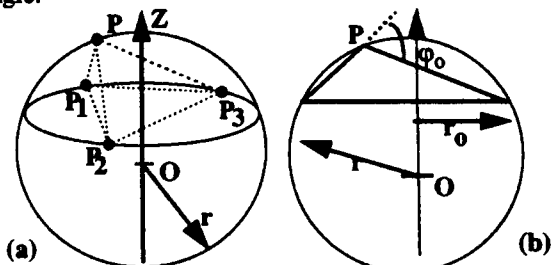


Figure 4: Definition of the Simplex Angle

It is clear that the simplex angle is invariant by rotation, translation, and scaling. In the remainder of the paper, the simplex angle ϕ_0 at a node P will be denoted by $g(P)$. It is important to note that other definitions of $g(P)$ are possible as long as the definition guarantees that $g(P)$ is invariant by rigid transformations and by scaling. We selected this definition because it is easy to compute and it is reasonably stable with respect to small variation of the surface.

3.4 Simplex Angle Image

We have extended the notions of regularity and simplex angle to three-dimensional surfaces; we can now extend the circular representation developed in two dimensions to a spherical representation in three dimensions. Let \mathcal{M} be a mesh of points on a surface such that it has the topology of the quasi-regular mesh of Section 3.1. Let \mathcal{S} be a reference mesh with the same number of nodes on the unit sphere. We can establish a one-to-one mapping h between the nodes of \mathcal{M} and the nodes of \mathcal{S} . The mapping h depends only on the topology of the mesh and the number of nodes. Specifically, for a given size of the mesh, $M = 20 \times N^2$, where N is the frequency of the mesh (Section 3.1), we can define a canonical numbering of the nodes that represents the topology of any M -mesh. In other words, if two nodes from two different M -mesh have the same index, so do their neighbors. With this indexing system, $h(P)$, where P is a node of the spherical mesh, is the node of the object mesh that has the same index as P .

Given h , we can store at each node P of \mathcal{S} the simplex angle of the corresponding node on the surface $g(h(P))$. The resulting structure is a quasi-regular mesh on the unit sphere, each node being associated with a value corresponding to the simplex angle of a point on the original surface. By analogy with the EGI, we call this representation the Simplex Angle Image (SAI). In the remainder of the paper, we will denote by $g(P)$ instead of $g(h(P))$ the simplex angle associated with the object mesh node $h(P)$ since there is no ambiguity.

If the original mesh \mathcal{M} satisfies the condition of local regularity, then the corresponding SAI has several important properties. First, the SAI is invariant by translation and scaling of the original object, given a mesh \mathcal{M} . This condition is satisfied because the simplex angle itself is invariant by translation and scaling (Section 3.4), and because \mathcal{M} still satisfies the local regularity condition after translation and scaling (Section 3.2).

From this definition of the mapping h , we can easily see the origin the property of connectivity conservation mentioned in the Introduction. If two nodes P_1 and P_2 are connected on the spherical mesh, then the two corresponding nodes $M_1 = h(P_1)$ and $M_2 = h(P_2)$ on the object mesh are also connected by an arc of the object mesh. The property holds because of the definition of h which depends only on the topology of the mesh, not on the positions of the nodes.

The fundamental property of the SAI is that it represents an object unambiguously up to a rotation. More precisely, if \mathcal{M} and \mathcal{M}' are two meshes on the same object with the same number of nodes both satisfying the local regularity condition, then the corresponding SAIs \mathcal{S} and \mathcal{S}' are identical up to a rotation of the unit sphere. This property holds even in the case of arbitrary non-convex objects because of the connectivity conservation property. Strictly speaking, this is true only as the number of nodes becomes very large because the nodes of one sphere do not necessarily coincide with the nodes of the rotation version of the other sphere. (This problem is addressed in Section 5.1.)

4.0 Building SAIs from 3-D Data

In the previous sections, we have defined the notion of locally regular mesh and its associated SAI. In this section, we describe the algorithm for computing such a

mesh from an input object. We assume that we have some input description of an object. The only requirement is that the input description allows for computing the distance between an arbitrary point in space and the surface of the object.

The general approach is to first define an initial mesh near the object (Section 4.2) and to deform it by moving its nodes until the mesh satisfies two conditions (Section 4.1): It must be close to the input object, and it must satisfy the local regularity condition. Once the mesh is created, the simplex angle is computed at every node and is mapped on the unit sphere (Section 4.3).

4.1 Mesh Deformation

The formalism of deformable surfaces [8] is applied to deform the mesh. Specifically, each node is subject to two types of forces. The first type of forces brings a node closer to the input surface, while the second type forces the node to satisfy the normal constraint. Let F_o be the force of the first type applied at a given node N , and F_g be the force of the second type at the same node. If P_{t+1} , P_t , and P_{t-1} are the positions of node P at three consecutive iterations, the update rule is defined as:

$$P_{t+1} = P_t + F_o + F_g + D(P_t - P_{t-1})$$

This expression is simply the discrete version of the fundamental equation describing a mechanical system subject to two forces and to a damping coefficient D . In practice, the iterative update of the mesh is halted when the relative displacements of the nodes from one iteration to the next are small.

F_o is defined by calculating the point P_c from the original surface that is closest to the node, that is: $F_o = kPP_c$, where k is the spring constant of the force which must be between 0 and 1.

The curvature force F_g is calculated by computing the point P_g that is on the line normal to the triangle formed by the three neighbors of P and containing G (Figure 3), and such that the mesh curvature at P and P_g are the same: $g(P_g) = g(P)$. Those two conditions ensure that P_g satisfies the local regularity condition while keeping the original mesh curvature. F_g is defined as a spring force proportional to the distance between P and P_g : $F_g = aPP_g$

4.2 Initialization

For the iterative mesh update to converge, the mesh must be initialized to some shape that is close to the initial shape. We use two different approaches depending on whether the input data is measured on the object by a sensor, or a synthetic CAD model.

In the case of sensor data, we use the techniques presented in [8] using deformable surfaces to build a triangulated mesh that approximates the object. Once a triangulation is obtained, the mesh is initialized by tessellating the ellipsoid of inertia of the input shape. Although the ellipsoid is only a crude approximation of the object, it is close enough for the mesh deformation process to converge.

In the case of a synthetic CAD model as input, for example a polyhedron, the ellipsoid of inertia is computed directly from the synthetic model. A regular mesh is mapped on the ellipsoid in the same manner as in the previous case.

Once the initial ellipsoid is generated, the mesh generation is completely independent of the actual format of the input data. The only data-dependent operation is the computation of the object point closest to a given node.

4.3 From Mesh to SAI

Once a regular mesh is created from the input data, a reference mesh with the same number of nodes is created on the unit sphere. The value of the angle at each node of the mesh is stored in the corresponding node of the sphere. The SAI building algorithm is illustrated in Figure 6 with range data as input and in Figure 8 with a polyhedral model as input. Figure 5 shows three views of a green pepper from which three 240x256 range images were taken using the OGIS range finder. The images are merged and an initial description of the object is produced using the deformable surface algorithm. Figure 6 (a) and Figure 6 (b) show the initial mesh mapped on the ellipsoid and the mesh at an intermediate stage. Figure 6 (c) shows the final regular mesh on the object. Figure 6 (d) shows the corresponding SAI. The meshes are displayed as depth-cued wireframes. The SAI is displayed by placing each node of the sphere at a distance from the origin that is proportional to the angle stored at that node.

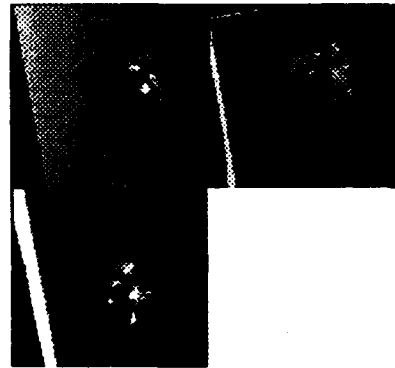


Figure 5: Three Views of an Object

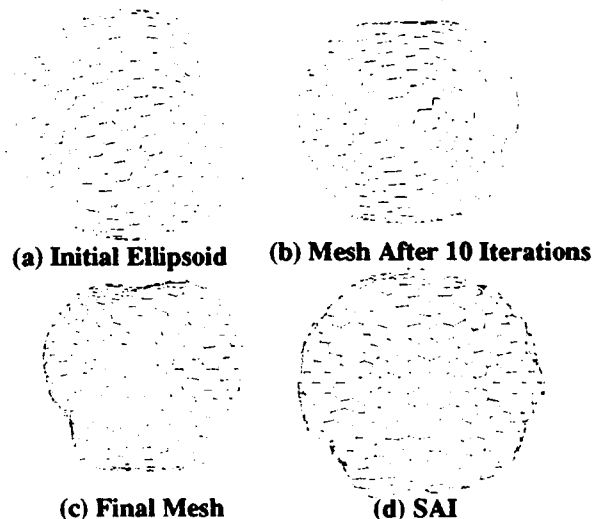


Figure 6: Building SAI from Range Data

Figure 8 (a) and Figure 8 (b) show the mesh and the SAI, respectively, in the case of an object initially described as a polyhedron as generated by the VANTAGE CAD system shown in Figure 7. The arrow between Figure 8 (a) and Figure 8 (b) shows the correspondence between object mesh and its SAI. The vertical

crease in the middle of the SAI corresponds to the concave region between the two cylinders. The top and bottom regions of the SAI exhibit large values of the angle corresponding to the transition between the cylindrical and planar faces at both extremities of the object. In this example, the SAI exhibits some noise in regions that are near the edges between faces of the object. In practice, the SAI is smoothed before being used for recognition.

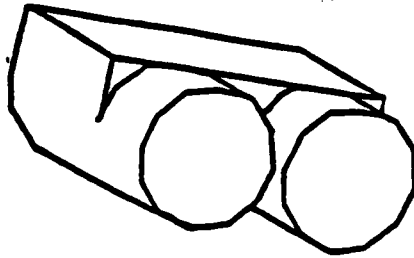


Figure 7: Geometric Model of an Object

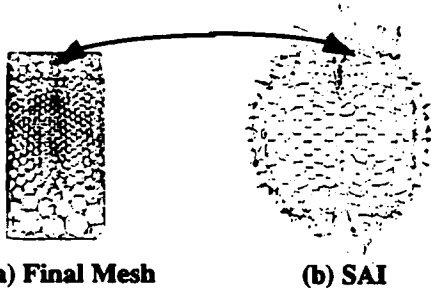


Figure 8: Building the SAI from a Polyhedral Model

5.0 Matching Objects

We now address the matching problem: Given two SAIs, determine whether they correspond to the same object and find the rigid transformation between the two instances of the object. As discussed in Section 3.0, the representations of a single object in two different poses are related by a rotation of the underlying sphere. Therefore, the most straightforward approach is to compute a distance measure between the SAIs and to find the rotation yielding minimum distance. The full 3-D transformations can be computed based on this rotation. This approach is expensive because it requires the testing of the entire 3-D space of rotations. We discuss strategies to reduce the search space in Section 5.3. Before that, in Sections 5.1 and 5.2, we discuss the distance measure and the computation of the final rigid transformation, respectively.

5.1 Finding the Best Rotation

Let S and S' be the spherical representations of two objects. Denoting by $g(P)$, *resp.* $g'(P)$, the value of the simplex angle at a node P of S , *resp.* P of S' , S and S' are representations of the same object if there exists a rotation R such that $g'(P) = g(RP)$ for every point P of S' . Since the SAI is discrete, $g(RP)$ is not defined because in general RP will fall between nodes of S . We define a discrete approximation of $g(RP)$, $G(RP)$, by interpolating the values of g at the four nodes of S nearest to RP .

The problem now is to find this rotation using the discrete representation of S and S' . This is done by defining a distance $D(S, S', R)$ between SAIs as the sum of squared

differences between the simplex angles at the nodes of one of the sphere and at the nodes of the rotated sphere:

$$D(S, S', R) = \sum_S (g'(P) - G(RP))^2$$

The minimum of D corresponds to the best rotation that brings S and S' in correspondence.

It is important to note that the rotation is *not* the rotation between the original objects; it is the rotation of the *spherical representations*. An additional step is needed to compute the actual transformation between objects as described below.

5.2 Computing the Full Transformation

The last step in matching objects is to derive the transformation between the actual objects, given the rotation between their SAIs. The rotational part of the transformation is denoted by R_o , the translational part by T_o . Given a SAI rotation R , for each node P' of S' we compute the node P of S that is nearest to RP' . Let M , *resp.* M' , be the point on the object corresponding to the node P of S , *resp.* P' of S' . A first estimate of the transformation is computed by minimizing the sum of the squared distances between the points M of the first object and the corresponding points $R_o M' + T_o$ of the second object. The resulting transformation is only an approximation because it assumes that the nodes from the two meshes correspond exactly. We use an additional step to refine the transformation by looking for the node M closest to M' for every node of the mesh and by computing again the optimal pose.

5.3 Reducing the Search Space

As mentioned in Section 5.1, the brute force approach to finding the best mesh rotation is too expensive to be practical. However, several strategies can be used to make it more efficient. The first strategy is to use a coarse-to-fine approach to locating the minimum of the function D of . In this approach, the space of possible rotations, defined by three angles of rotation about the three axis, (ϕ, θ, ψ) , is searched using large angular steps $(\Delta\phi, \Delta\theta, \Delta\psi)$. After this initial coarse search, a small number of locations are identified around which the minimum may occur. The space of rotations is again searched around each potential minimum found at the coarse level using smaller angular steps $(\delta\phi, \delta\theta, \delta\psi)$. Typical values are $\Delta\phi = \Delta\theta = \Delta\psi = 10^\circ$. More levels of search may be more efficient, although we have not yet tried to determine the best combination of coarse-to-fine levels.

The second approach is to use a priori knowledge about the relative poses of the objects to reduce the search space. For example, the rotation defined by the axis of inertia of the SAIs can be used as a starting point for the search. In practice, using the axis of inertia is very effective in pruning the search space as long as the visible part of the object is large enough.

5.4 Example

Figure 9 shows three views of the same object as in Figure 6 placed in a different orientation. A model is built from the three corresponding range images using the approach described in 4.3. Figure 12 illustrates the difference of pose between the two models computed from the two sets of images. Figure 10 shows the value

of the SAI distance measure. The distance measure is displayed as a function of ϕ and θ . The displayed value at (ϕ, θ) is the minimum value found for all the possible values of ψ . This display shows that there is a single sharp minimum corresponding to the rotation that brings the SAI in correspondence. Figure 11 shows one of the models backprojected in the image of the other using the transformation computed from the SAI correspondence using the algorithms of Section 5.2. Figure 11 (a) is the original image; Figure 11 (b) is the backprojected model. To illustrate the accuracy of the registration, Figure 12 (a) shows the superimposition of the cross sections of the two models in the plane before matching. Figure 12 (b) shows the same cross sections after matching. These displays show that the transformation is correctly computed in that the average distance between the two models after transformation is on the order of the accuracy of the range sensor.



Figure 9: Three Views of the Object of Figure 6

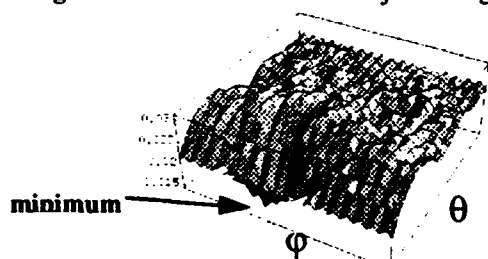


Figure 10: Distance Between SAIs as Function of (ϕ, θ)

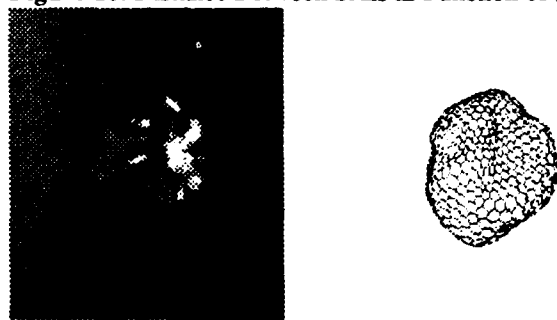


Figure 11: Display of the Model in the Computed Pose

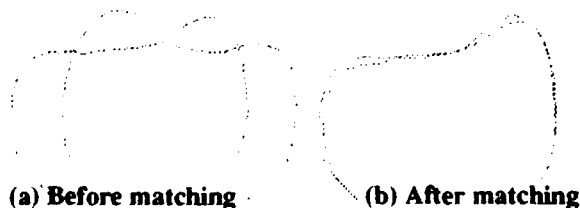


Figure 12: Relative Positions of the Two Objects

6.0 Partial Views and Occlusion

Up to now we have assumed that we have a complete model of the object, as in Figure 8, or that we have data covering the entire surface of the object, as in Figure 6. This assumption is appropriate for building reference models of objects. During the recognition phase, however, only a portion of the object is visible in the scene. The matching algorithm of Section 5.0 must be modified to allow for partial representations. The algorithm used for extracting the initial surface model is able to distinguish between regions of the mesh that are close to input surfaces or to data points, and parts that are interpolated between input data. The first type of region is the visible part of the mesh, and the second type is the occluded part of the mesh.

The situation is illustrated in Figure 14 in the case of a two-dimensional contour. In Figure 14 (a) a contour is approximated by a mesh of eight points. The mesh is assumed to be regular, that is all the points of the mesh are equidistant. Let $L = 8l$ be the total length of the mesh. Figure 14 (b) shows the same contour with one portion hidden. The occluded portion is shown as a shaded curve. The visible section is approximated by a regular mesh of eight nodes of length $L_1 = 8l_1$. Since the occluded part is interpolated as a straight line, the length of this mesh is smaller than the total length of the mesh on the original object: $L > L_1$. Conversely, the length of the part of the representation corresponding to the visible part, L_2 shown in Figure 14 (d), is greater than the length of the same section of the curve on the original representation, L^* shown in Figure 14 (c). In order to compute the distance measure of Section 5.0, the SAI of the observed curve must be scaled so that it occupies the same length on the unique circle as in the reference representation of the object. If L^* were known, the scale factor would be $k = L^*/L_2$. In reality, L^* is not known because we do not yet know which part of the reference curve corresponds to the visible part of the observed curve. To eliminate L^* , we use the relation $L_1/L = L^*/2\pi$. This relation simply expresses the fact that the ratios of visible and total length in object and representation spaces are the same, which is always true when the mesh is regular. L^* can be eliminated from these two relations, yielding an expression of $k = 2\pi L_1/L_2L$.

The situation is the same in three dimension except that lengths are replaced by areas A, A_1, A_2, A^* . The previous expression becomes $k = 4\pi A_1/A_2A$.

The direct extension from two to three dimension is only an approximation because the relation $A_1/A = A^*/4\pi$, holds only if the area per node is constant over the entire mesh. In practice, however, the area per node is nearly constant for a mesh that satisfies the local regularity condition.

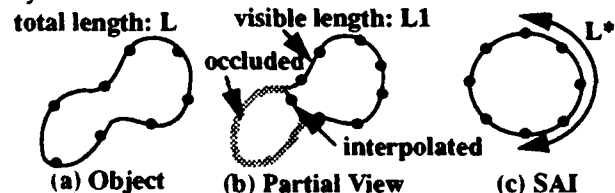


Figure 13: Matching Partial Representation in 2-D

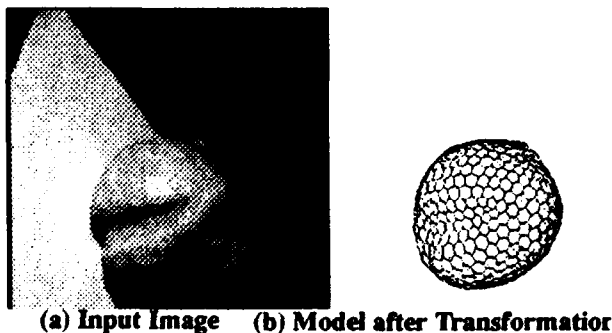
Once k is computed, the appropriate scaling is applied to the SAI by moving the nodes on the surface of

the sphere given the scaling ratio k . After scaling, the distribution of nodes on the part of the SAI corresponding to the visible part of the object in the scene and the distribution in the corresponding region of the model SAI are identical. The key in this algorithm is the connectivity conservation property of the SAI mentioned previously.

We now show two examples of recognition in the presence of occlusion. In the first example, a range image of an isolated object is taken. Then a complete model of the object is matched with the SAI representation from range data. Figure 15 shows the model backprojected in the observed image using the computed transformation. In this example, the reference model was computed by taking three registered range images of the object as in the example of Figure 6. Only about 30% of the object is visible in the image. The remaining 70% of the representation built from the image is interpolated and is ignored in the estimation of the SAI distance. Figure 17 displays the graph of the distance between SAIs as function of rotation angles. Figure 16 shows two views of the distance as a function of ϕ and θ . Figure 17 shows the same function displayed in ϕ - ψ space. These displays demonstrate that there is a well-defined minimum at the optimal rotation of the SAIs.

In the second example, the reference model is the CAD model of Figure 8. The result of the matching is shown in Figure 18 in which the model is displayed in the orientation computed from the range data using the SAI matching. Only part of the object is visible in the image because of self occlusion and because of occlusion from other objects in the scene.

In both examples, the deformable surface algorithm is used to separate the object from the rest of the scene and to build an initial surface model, as described in [8]. If there is no data point in its vicinity, the visible portion of the object mesh and the corresponding SAI are identified by marking a node of the mesh as interpolated. To illustrate the effect of scaling, Figure 19 (a) shows the SAI computed from the image of Figure 15, Figure 19 (b) shows the SAI after the scaling in applied to compensate for occlusions. The density of points increases in the region that corresponds to the visible part of the object (indicated by the solid arrow). Conversely the density of points decreases in the region corresponding to the occluded part of the object (indicated by the shaded arrow). These examples show that the SAI matching algorithm can deal with occlusions and partial views, even when only a relatively small percentage of the surface is visible.



(a) Input Image (b) Model after Transformation
Figure 14: Model Registered with Input Image

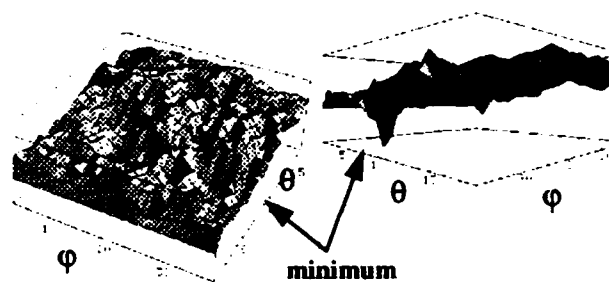


Figure 15: Distance Between as Function of ϕ and θ

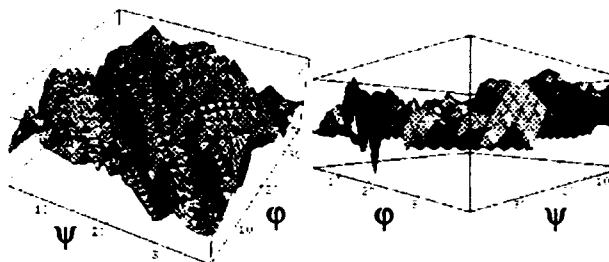
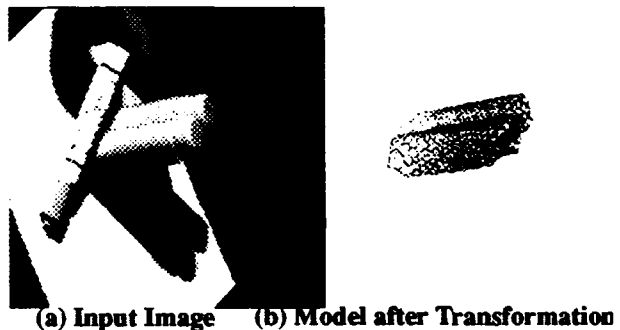
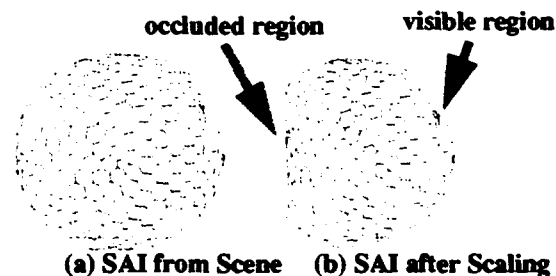


Figure 16: Distance Between as Function of ϕ and ψ



(a) Input Image (b) Model after Transformation
Figure 17: Model Registered with Input Image



(a) SAI from Scene (b) SAI after Scaling
Figure 18: Effect of Occlusion-Compensating Scaling

7.0 Conclusion

In this paper, we introduced a new approach for building and recognizing models of curved objects. The basic representation is a mesh of nodes on the surface that satisfies certain regularity constraints. We introduced the notion of simplex angle as a curvature indicator stored at each node of the mesh. We showed how a mesh can be mapped into a spherical representation in canonical manner, and how objects can be recognized by computing the distance between spherical representations.

The SAI representation has many desirable properties that make it very effective as a tool for 3-D object recognition. The SAI is invariant with respect to translation, rotation, and scaling of the object. This invariance allows the recognition algorithm to compare shapes through the computation of distances between SAI's without requiring explicit matching between object features or explicit computation of object pose. The SAI preserves connectivity between parts of the object in that nodes that are neighbors on the object mesh are also neighbors on the SAI. Thus the SAI can handle non-convex objects, partial views, and occluded objects thanks to the property of connectivity conservation of the SAI.

Results show that the SAI representation can be used to determine the pose of an object in a range image. This approach is particularly well suited for applications dealing with natural objects. Typically, conventional object modeling and recognition techniques would not work due to the variety and complexity of natural shapes.

Many issues remain to be addressed. First, we need to improve the search for the minimum distance between SAI's during the recognition phase. This improvement can be achieved by improving the coarse-to-fine approach to extrema localization, and by using cues computed from the original data to restrict the area in which the extrema are searched. Another important extension is the use of appearance information such as hue or albedo, in addition to geometric information. Such information can be included in the SAI representation by storing appearance data at every node of the SAI in addition to the curvature measure. Appearance data can be incorporated in the distance measure between SAI's to help disambiguate between shapes that are similar geometrically but have different appearance features. We are now pursuing such an approach using color images.

8.0 References

- [1] Aleksandrov, A.D., Zalgaller, V.A., "Intrinsic Geometry of Surfaces", Translation of Mathematical Monographs Series, AMS Publisher, 1967
- [2] Besl, P.J., Kay, N.D., "A Method for Registration of 3-D Shapes", PAMI-14(2), 1992
- [3] Besl, P., Jain, R., "Segmentation Through Symbolic Surface Descriptions", Proc. ICCV'86, pp 77-85, 1986
- [4] Bobick, A.F., Bolles, R.C., "The Representation Space Paradigm of Concurrent Evolving Object Descriptions", PAMI, Vol. 14, No. 2, p. 146, 1992
- [5] Bolle, R.M., B.C. Vemuri, "Three Dimensional Surface Reconstruction Methods", PAMI, Vol. 13, pp. 1-14, 1991.
- [6] Brady, M. and Ponce, J. and Yuille, A. and Asada, H., "Describing surfaces", Proc. 2nd International Symposium on Robotics, MIT Press, Cambridge, MA, 1985
- [7] Brou, P., "Using the Gaussian image to find the orientation of object", IJRR, Vol. 3, No. 4, pp. 89-125, 1983
- [8] Delingette, H., Hebert, M. and Ikeuchi, K., "Shape Representation and Image Segmentation Using Deformable Surfaces", Image and Vision Computing 10 (3), April 1992
- [9] Faugeras, O.D. and Hebert, M., "The representation, recognition, and locating of 3-D objects", IJRR, Vol. 5, No. 3, pp. 27-52, 1986
- [10] Ferrie, F.P., Lagarde, J. and Whaite, P., "Darboux Frames, Snakes, and Super-Quadrics: Geometry from the Bottom-Up", Proc. IEEE Workshop on Interpretation of 3D Scene, pp. 170-176, 1989,
- [11] Forsyth, D.A., Mundy, J.L., Zisserman, A., Coelho, C., Heller, A., Rothwell, C., "Invariant Descriptors for 3-D Object Recognition and Pose", PAMI, Vol. 13, October 1992
- [12] Grimson, W. E. L. and Lozano-Perez, T., "Localizing Overlapping Parts by Searching the Interpretation Tree", PAMI-9(4), July 1987
- [13] Horn, B.K.P., "Extended Gaussian Images", Proc. of the IEEE, 72(12), December 1984
- [14] Ikeuchi, K., "Recognition of 3-D objects using the extended Gaussian image", International Joint Conf. on Artificial Intelligence, 595-600, 1981
- [15] Ikeuchi, K. and Hong, K.S., "Determining Linear Shape Change: Toward Automatic Generation of Object Recognition Program," CVGIP:IU, 53(2), pp.154-170, March 1991
- [16] Kang, S.B. and Ikeuchi, K., "Determining 3-D Object Pose using the Complex Extended Gaussian Image," Proc. CVPR-91, Hawaii, June 1991
- [17] Little, J.J., "Determining object attitude from extended Gaussian images", Proc. of 9th Intern. Joint Conf. on Artificial Intelligence, 960-963, 1985
- [18] Loeb, A.L., "Space Structures", Addison-Wesley, 1976
- [19] Lowe, D.G., "Three-dimensional object recognition from single two-dimensional images", Artificial Intelligence, 31, 1987, 355-395
- [20] Oshima, M. and Shirai, Y., "Object recognition using three-dimensional information", PAMI-5(4), July 1983
- [21] Pentland, A. P., "Perceptual Organization and the Representation of Natural Form", Artificial Intelligence, 28, 2, 293-331, 1986
- [22] Taubin, G., "Recognition and Positioning of Rigid Objects Using Algebraic and Moment Invariants", PhD Dissertation, Brown University, 1990
- [23] Taubin, G., Cukierman, F., Sullivan, S., Ponce, J., and Kriegman, D.J., "Parametrizing and Fitting Bounded Algebraic Curves and Surfaces", Proc. Computer Vision and Pattern Recognition, June 1992.
- [24] Wenninger, M., "Polyhedron Models", Cambridge University Press, London, 1971
- [25] Wenninger, M., "Dual Models", Cambridge University Press, London, 1983

Statistical Object Recognition with the Expectation-Maximization Algorithm in Range-Derived Features

William M. Wells III *†

Massachusetts Institute of Technology

Artificial Intelligence Laboratory

545 Technology Square, Cambridge, Massachusetts 02139

sw@ai.mit.edu

Abstract

An extension to the alignment approach is proposed that includes a pose refinement step before verification. In the alignment approach the pose estimates of the initial hypotheses tend to be somewhat inaccurate, since they are based on minimal sets of corresponding features. A method is described that refines the pose estimate while simultaneously identifying and incorporating the constraints of all supporting image features. The strategy also makes practical initial alignments based on low resolution features – which, being less numerous, allow faster running times.

Two statistical formulations of model-based recognition are described: MAP Model Matching, and Posterior Marginal Pose Estimation (PMPE). These formulations use a normal model for feature fluctuations. Empirical evidence is provided from the domain of video edge features indicating that normal probability densities are good models of feature fluctuations – better than uniform densities in that domain. The evidence is provided in the form of observed and fitted cumulative distributions. The results of some statistical tests are reported.

The Expectation – Maximization (EM) algorithm is discussed as a method of carrying out local searches in pose space of the PMPE objective function. A recognition experiment is described where the method is used with fea-

tures derived from synthetic range images.

1 Introduction

1.1 The Statistical Approach and Object Recognition

In this paper, visual object recognition is approached via the principles of maximum-likelihood (ML) and maximum-a-posteriori probability (MAP). These principles, along with specific probabilistic models of aspects of object recognition, are used to derive objective functions for evaluating and refining recognition hypotheses. The ML and MAP criteria have a long history of successful application in formulating decisions and in making estimates from observed data. They have attractive properties of optimality, and are often useful when measurement errors are significant.

In other areas of computer vision, statistics has proven useful as a theoretical framework. The work of Yuille, Geiger and Bülthoff on stereo [1] is one example, while the work of Geman and Geman on image restoration [2] is another. The statistical approach that is used in this paper converts the recognition problem into a well defined (although not necessarily easy) optimization problem. This has the advantage of providing an explicit characterization of the problem, while separating the specification of the problem from the description of the algorithms used to attack it. Ad-Hoc objective functions have been profitably used in some areas of computer vision. Such an approach is used by Barnard in stereo matching [3], Blake and Zisserman [4] in image restoration and Beveridge, Weiss and Riseman [5] in line segment based model matching. With this approach, plausible forms for components of the objective function are often combined using trade-off parameters. Such trade-off parameters are determined empirically. An advantage of deriving objective functions from statistical theories is that the assumptions become explicit in that the forms of the objective function components are clearly related to specific probabilistic models. A second advantage is that the trade-off parameters in the objective function can be derived from measurable statistics of the domain.

1.2 Alignment

The basic strategy of the alignment method of recognition [6] is to use separate mechanisms for generating and

*The author is currently affiliated with Harvard Medical School, Brigham and Women's Hospital, Department of Radiology, Surgical Planning Laboratory.

†This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by an Office of Naval Research University Research Initiative grant under contract N00014-86-K-0685, and in part by the Advanced Projects Agency of the Department of Defense under Army contract number DACA76-85-C-0010 and under Office of Naval Research contract N00014-85-K-0124. Summer support was provided by Group 53 at the Massachusetts Institute of Technology, Lincoln Laboratory.

testing hypotheses.

Recently, indexing methods have become available for efficiently generating hypotheses in recognition. These methods avoid a significant amount of search by looking up, in pre-computed tables, the object features that might correspond to a group of image features. The geometric hashing method of Lamdan and Wolfson [7] uses invariant properties of small groups of features under affine transformations as the lookup key. Clemens and Jacobs [8] describe an indexing method that gains efficiency by using a feature grouping process to select small sets of image features that are likely to belong to the same object in the scene.

1.3 Align - Refine - Verify

The recognition strategy advocated in this work may be summarized as "align-refine-verify." The key observation is that local search in pose space may be used to refine the hypothesis from the alignment stage before verification is carried out. With the alignment method, the pose estimates of the initial hypotheses tend to be somewhat inaccurate, since they are based on minimal sets of corresponding features. Better pose estimates (hence, better verifications) are likely to result from using all supporting image feature data, rather than a small subset. This paper describes a method that refines the pose estimate while simultaneously identifying and incorporating the constraints of all supporting image features.

The strategy also makes practical initial alignments based on lower resolution features. Using low resolution features in indexing is attractive, because there are fewer features at low resolution, allowing faster running times.

1.4 Structure of the Paper

In Sections 2 and 3 probabilistic models of feature correspondences and feature fluctuations are described, while Section 4 discusses linear models of feature projection. Section 5 describes a statistical method of simultaneously evaluating correspondences and object pose. Building on this, Section 6 outlines a statistical method of object pose evaluation: Posterior Marginal Pose Estimation (PMPE). Section 7 describes the use of the expectation - maximization (EM) algorithm for solving the PMPE objective function. An experiment using the EM algorithm with features derived from synthetic range images is presented in Section 8, and related work is discussed in Section 9.

2 Modeling Feature Correspondence

Let the image that is to be analyzed be represented by a set of v -dimensional point features

$$Y = \{Y_1 Y_2 \dots Y_n\}, \quad Y_i \in R^v.$$

Image features are discussed in more detail in Sections 3 and 4.

The object to be recognized is also described by a set of features, these are represented by real matrices:

$$M = \{M_1 M_2 \dots M_m\}.$$

Additional details on object features appears in Section 4.

The interpretation of the features in an image is represented by the variable Γ , which describes the mapping from image features to object features. This is also referred to as the *correspondences*.

$$\Gamma = \{\Gamma_1 \Gamma_2 \dots \Gamma_n\}, \quad \Gamma_i \in M \cup \{\perp\}.$$

In an interpretation, each image feature, Y_i , will be assigned either to some object feature M_j , or to the background, which is denoted by the symbol \perp . This symbol plays a role similar to that of the null character in the interpretation trees of Grimson and Lozano-Pérez [9]. Each variable Γ_i represents the assignment of the corresponding image feature Y_i , it may take on as value any of the object features M_j , or the background, \perp .

2.1 Independent Correspondence Model

In this section a simple probabilistic model of correspondences is described. The intent is to capture some information bearing on correspondences before the image is compared to the object. This model has been designed to be a reasonable compromise between simplicity and accuracy.

In this model, the correspondence status of differing image features are assumed to be independent, so that

$$p(\Gamma) = \prod_i p(\Gamma_i). \quad (1)$$

There is evidence against using statistical independence here, for example, occlusion is locally correlated. Independence is used as an engineering approximation that simplifies the resulting formulations of recognition. It may be justified by the good performance of the recognition experiments described in Section 8. Few recognition systems have used non-independent models of correspondence. Breuel described one approach [10]. In [11] the independence assumption of Equation 1 has been relaxed in a Markov Random Field (MRF) model of correspondences that is meant to capture the correlated aspects of occlusion.

The component probability function is designed to characterize the amount of clutter in the image, but to be otherwise as non-committal as possible:

$$p(\Gamma_i) = \begin{cases} B & \text{if } \Gamma_i = \perp \\ \frac{1-B}{m} & \text{otherwise} \end{cases} \quad (2)$$

The joint model $p(\Gamma)$ is the maximum entropy probability function that is consistent with the constraint that the probability of an image feature belonging to the background is B . B may be estimated by taking simple statistics on images from the domain.

The independent correspondence model is used in the experiments described in this paper.

3 Modeling Image Features

3.1 Uniform Model for Background Features

Image features belonging to the background are assumed to be uniformly distributed throughout the v -dimensional coordinate space of the image,

$$p(Y_i | \Gamma, \beta) = \frac{1}{W_1 \dots W_v} \quad \text{if } \Gamma_i = \perp \quad (3)$$

(The PDF is zero outside the coordinate space of the image.) The position and orientation, or *pose* of the object is described by β . The W_i are the dimensions of the feature coordinate space.

Equation 3 describes the maximum entropy PDF consistent with the constraint that the coordinates of image features are always expected to lie within the space of the image.

3.2 Normal Model for Matched Features

Image features that are matched to object features are assumed to be normally distributed about their predicted position in the image,

$$p(Y_i | \Gamma, \beta) = G_{\psi_{ij}}(Y_i - \mathcal{P}(M_j, \beta)) \quad \text{if } \Gamma_i = M_j \quad (4)$$

$G_{\psi_{ij}}$ is the v -dimensional Gaussian probability density function with covariance matrix ψ_{ij} : $G_{\psi_{ij}}(x) = (2\pi)^{-\frac{v}{2}} |\psi_{ij}|^{-\frac{1}{2}} \exp(-\frac{1}{2} x^T \psi_{ij}^{-1} x)$. The covariance matrix ψ_{ij} is discussed more fully in Section 3.3.

When $\Gamma_i = M_j$, the predicted coordinates of image feature Y_i are given by $\mathcal{P}(M_j, \beta)$, the projection of object feature j into the image with object pose β . Projection and pose are discussed in more detail in Section 4.

3.2.1 Empirical Evidence for Normal Model of Feature Fluctuations

This section describes some empirical evidence from the domain of video image edge features indicating that, in that domain, normal probability densities are good models of feature fluctuations, and that they are better than uniform probability densities. The evidence is provided in the form of observed and fitted cumulative distributions. Additionally, the results of some statistical tests are reported.

The data that is analyzed are the perpendicular deviations of edge features derived from video images. The features are shown in Figure 1.

The model features are from mean edge images (these are averaged with respect to illumination), and the edge operator used in obtaining the image features is ridges in the magnitude of the image gradient. These edge detection methods are described in [11]. The smoothing standard deviation used in the edge detection was 1.93 pixels. The object is at the same pose in the model and image scenes, the variation in features is due to changes in lighting, the presence of background features in the "image" scene and the vagaries of edge detection.

For the analysis in this section, the feature data consists of the average of the X and Y coordinates of the pixels from edge curve fragments. The features are displayed as circular arc fragments for clarity. The edge curves were broken arbitrarily into 10 pixel fragments.

Correspondences from image features to model features were established by a neutral subject using a mouse. These correspondences are indicated by heavy lines in Figure 2. Perpendicular and parallel deviations of the corresponding features were calculated with respect to the normal vectors to edge curves at the image features.

Figure 3 shows the cumulative distribution of the perpendicular deviations. The cumulative distribution of

a fitted normal density is plotted in the left figure as heavy dots over the observed distribution. The distribution was fitted to the data using the maximum-likelihood method. These figures show good agreement between the observed distribution, and the fitted normal distribution. The observed cumulative distribution is shown again on the right in Figure 3, this time with the cumulative distribution of a fitted uniform density over-plotted in heavy dots. As before, the uniform density was fitted to the data using the maximum-likelihood method. This figure shows relatively poor agreement between the observed and fitted distribution, in comparison to the normal density. Similar results obtained for the parallel deviations, and for a similar set of coarse features [11].

Kolmogorov-Smirnov tests were carried out [11] to evaluate the compatibility of the data with the fitted normal and uniform distributions. In the cases of fine perpendicular and parallel deviations, and coarse perpendicular deviations, refutation of the uniform model is strongly indicated. Strong contradictions of the fitted normal models are not indicated in any of the cases.

3.3 Oriented Stationary Statistics

The covariance matrix ψ_{ij} that appears in the model of matched image features in Equation 4 is allowed to depend on both the image feature and the object feature involved in the correspondence. Indexing on i allows dependence on the image feature detection process, while indexing on j allows dependence on the identity of the model feature. This is useful when some model features are known to be noisier than others. This flexibility is carried through the formalism of later sections. Although such flexibility can be useful, substantial simplification results by assuming that the features' statistics are stationary in the image. In its strict form this assumption may be too limiting. This section outlines a compromise approach, oriented stationary statistics, that was used in the experiments described in Section 8.

The Oriented Stationary Statistics method involves attaching a coordinate system to each image feature. The coordinate system has its origin at the point location of the feature, and is oriented with respect to the direction of the underlying curve at the feature point. When (stationary) statistics on feature deviations are measured, they are taken relative to these coordinate systems. When an image is presented for recognition, the constant feature covariance is specialized by rotating it to orient it with respect to each image feature. Oriented Stationary Statistics is described in more detail in [11].

4 Linear Projection Models

Pose determination is frequently framed as an optimization problem. The pose determination problem may be significantly simplified if the feature projection model is linear in the parameters of the transformation (the pose vector). The system described in this paper uses a projection model having this property, this enables solving the embedded optimization problem using least squares. Linear projection models may be written in the following

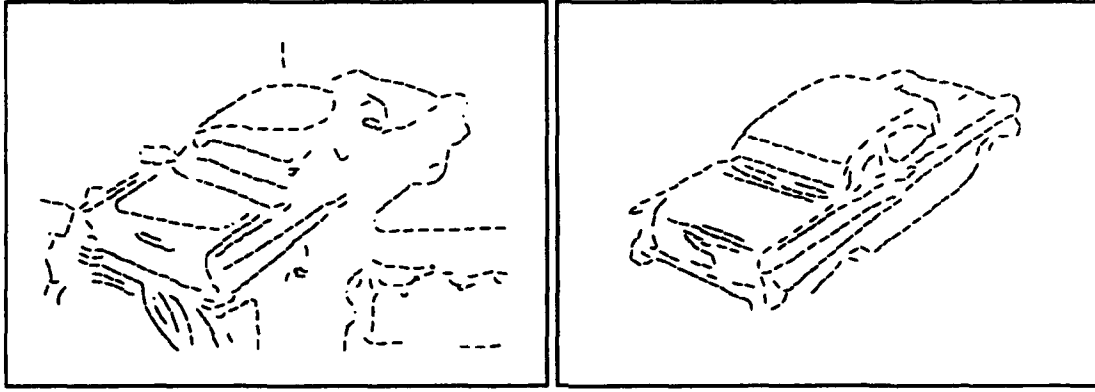


Figure 1: Image and Model Features

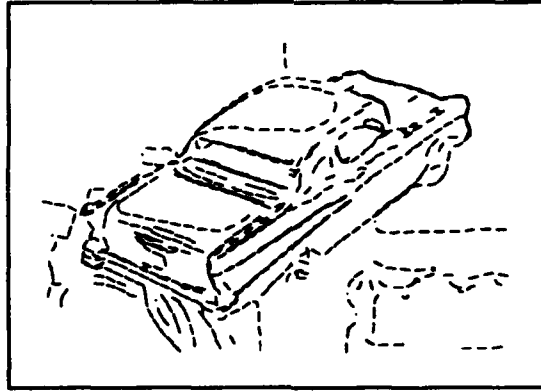


Figure 2: Feature Correspondences

form:

$$\eta_i = \mathcal{P}(M_i, \beta) = M_i \beta \quad (5)$$

The pose of the object is represented by β , a column vector, the object model feature by M_i , a matrix. η_i , the projection of the model feature into the image by pose β , is a column vector.

Several linear projection models were described in [12], and in [11].

4.1 2-D Oriented-Range Feature Model

A 2-D projection and feature model that incorporates local information about the coordinates, normal, and range at a point along a curve of range discontinuity, is defined by

$$\eta_i = \begin{bmatrix} p'_{ix} \\ p'_{iy} \\ c'_{ix} \\ c'_{iy} \end{bmatrix} \quad M_i = \begin{bmatrix} p_{ix} & -p_{iy} & 1 & 0 \\ p_{iy} & p_{ix} & 0 & 1 \\ c_{ix} & -c_{iy} & 0 & 0 \\ c_{iy} & c_{ix} & 0 & 0 \end{bmatrix}$$

$$\text{and} \quad \beta = \begin{bmatrix} \mu \\ \nu \\ t_x \\ t_y \end{bmatrix}$$

The coordinates of model point i are p_{ix} and p_{iy} . The coordinates of the model point i , projected into the image by pose β , are p'_{ix} and p'_{iy} . c_{ix} and c_{iy} are a vector whose direction is perpendicular to the range discontinuity and pointing away from the discontinuity, while the

length of the vector is the inverse of the range at the discontinuity. The counterparts in the image are given by c'_{ix} and c'_{iy} .

This transformation is equivalent to rotation by θ , scaling by s , and translation by T , where

$$T = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad s = \sqrt{\mu^2 + \nu^2} \quad \theta = \arctan\left(\frac{\nu}{\mu}\right)$$

The aggregate feature translates, rotates and scales correctly when used with imaging models where the object features scale according to the inverse of the distance to the object. This holds under perspective projection with attached range data when the object is small compared to the distance to the object.

This projection model was inspired by a method used by Faugeras and Ayache in their vision system HYPER [13], and it is used in the experiments described in Section 8.

5 MAP Model Matching

This section outlines MAP Model Matching¹, a means of evaluating joint hypotheses of match and pose using the MAP criterion.

Briefly, probability densities of image features, conditioned on the parameters of match and pose ("the parameters"), are combined with prior probabilities on the

¹An early version of this work appeared in [12] and [14].

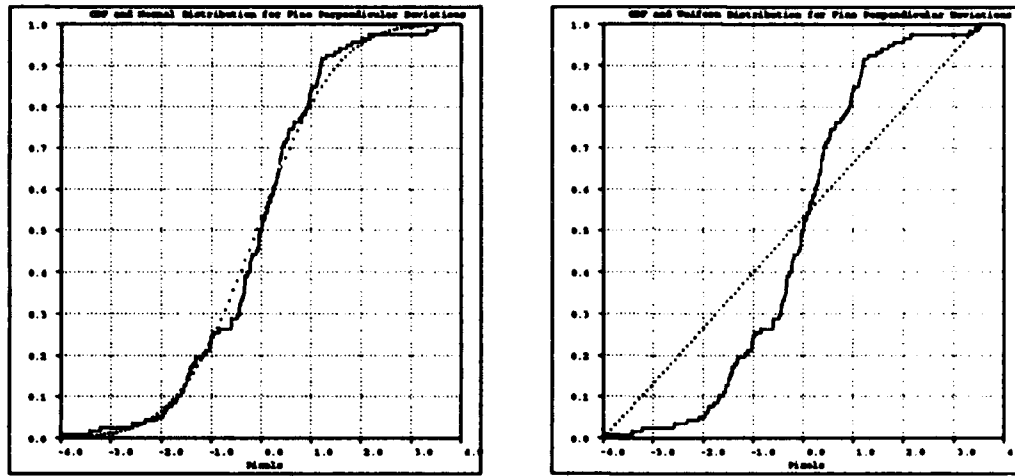


Figure 3: Observed Cumulative Distribution with Fitted Normal and Uniform Cumulative Distributions

parameters using Bayes' rule. The result is a posterior probability density on the parameters, given an observed image. An estimate of the parameters is then formulated by choosing them so as to maximize their posterior probability (Hence the term MAP). MAP estimators are especially practical when used with normal probability densities.

The parameters to be estimated in matching are the correspondences between image and object features, and the pose of the object in the image.

The probabilistic models of image features, conditioned on match and pose, that is described in Equations 3 and 4 may be written as follows:

$$p(Y_i | \Gamma, \beta) = \begin{cases} \frac{1}{W_1 W_2 \dots W_n} & \text{if } \Gamma_i = \perp \\ G_{\psi_{ij}}(Y_i - \mathcal{P}(M_j, \beta)) & \text{if } \Gamma_i = M_j \end{cases} \quad (6)$$

Here ψ_{ij} is the covariance matrix associated with image feature i and object model feature j . Thus image features arising from the background are uniformly distributed over the space of the image (the width of the image space along dimension i is given by W_i), and matched image features are normally distributed about their predicted locations in the image. In some applications ψ could be a constant - an assumption that the feature statistics are stationary in the image, or ψ may depend only on i , the image feature index. This is the case when the oriented stationary statistics model is used (see Section 3.3).

Assuming independent features, the joint probability density on image feature coordinates may be written as follows

$$\begin{aligned} p(Y | \Gamma, \beta) &= \prod_i p(Y_i | \Gamma, \beta) \\ &= \prod_{i: \Gamma_i = \perp} \frac{1}{W_1 W_2 \dots W_n} \times \\ &\quad \prod_{i: \Gamma_i = M_j} G_{\psi_{ij}}(Y_i - \mathcal{P}(M_j, \beta)) \end{aligned} \quad (7)$$

Next, a joint prior on correspondences and pose is constructed. Prior information on the pose is assumed to be supplied as a normal density,

$$p(\beta) = G_{\psi_\beta}(\beta - \beta_0) = (2\pi)^{-\frac{z}{2}} |\psi_\beta|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} x^T \psi_\beta^{-1} x\right) \quad (8)$$

Here ψ_β is the covariance matrix of the pose prior and z is the dimensionality of the pose vector, β . With this choice for the form of the pose prior the system is closed in the sense that the resulting pose estimate will also be normal. This is convenient for coarse-fine. If little is known about the pose a-priori, the prior may be made quite broad. This is expected to be often the case. If nothing is known about the pose beforehand, the pose prior may be left out. In that case the resulting criterion for evaluating hypotheses will be based on maximum-likelihood for pose, and on MAP for correspondences.

Assuming independence of the correspondences and the pose (before the image is compared to the object model), and using the independent correspondence model of Equations 1 and 2, a mixed joint prior probability function may be written as follows,

$$p(\Gamma, \beta) = G_{\psi_\beta}(\beta - \beta_0) \prod_{i: \Gamma_i = \perp} B_i \prod_{i: \Gamma_i \neq \perp} \frac{1 - B_i}{m}$$

The variable B has been generalized here to have an image feature dependent subscript, i . In the experiments reported in Section 8, $B_i = B$ for all i . This generalization is explored more fully in [11]. This probability function on match and pose is now used with Bayes' rule as a prior for obtaining the posterior probability of Γ and β :

$$p(\Gamma, \beta | Y) = \frac{p(Y | \Gamma, \beta) p(\Gamma, \beta)}{p(Y)} \quad (9)$$

where $p(Y)$ is the probability of the image being analyzed - a constant in terms of the parameters being estimated.

The MAP strategy is used to obtain estimates of the correspondences and pose by maximizing their posterior density with respect to Γ and β , as follows

$$\widehat{\Gamma, \beta} = \arg \max_{\Gamma, \beta} p(\Gamma, \beta | Y)$$

The search for maximizing joint hypotheses of match and pose may be ordered as either a search in correspondence space, or a search in pose space. Recognition experiments in correspondence based search were reported in [12], where heuristic beam search was used. In [11] it is shown that searching a specialization of this MAP criterion in pose space is equivalent to robust chamfer matching.

6 Posterior Marginal Pose Estimation

In the previous section the object recognition problem was posed as an optimization problem resulting from a statistical theory. In that formulation, a hypothesis was the position and orientation, or pose, of the object – as well as the correspondences between object and image features.

The formulation of recognition that is described in this section, Posterior Marginal Pose Estimation (PMPE)², builds on MAP model matching. It provides a smooth objective function for evaluating hypotheses that are simply the pose of the object. The pose is the most important aspect of the problem, in the sense that knowing the pose enables grasping or other interaction with the object.

PMPE was motivated by the observation that in heuristic searches over correspondences with the objective function of MAP model matching, hypotheses having implausible matches scored poorly in the objective function. Additional motivation was provided by the work of Yuille, Geiger and Bülthoff on stereo [1]. They discussed computing disparities in a statistical theory of stereo where a marginal is computed over matches.

Here we use the same strategy as MAP Model Matching for evaluating hypotheses now consisting only of pose – they are evaluated by their posterior probability, given an image: $p(\beta | Y)$. The posterior probability density of the pose may be computed from the joint posterior probability on pose and match, by formally taking the marginal over possible matches:

$$p(\beta | Y) = \sum_{\Gamma} p(\Gamma, \beta | Y)$$

In Section 5, Equation 9, $p(\Gamma, \beta | Y)$ was obtained via Bayes' rule from probabilistic models of image features, correspondences, and the pose. Substituting for $p(\Gamma, \beta | Y)$, the posterior marginal may be written as

$$p(\beta | Y) = \sum_{\Gamma} \frac{p(Y | \Gamma, \beta) p(\Gamma, \beta)}{p(Y)}$$

Using Equations 1 (the independent feature model) and 7, we may express the posterior marginal of β in

²An early version of this work appeared in [15].

terms of the component densities:

$$p(\beta | Y) = \frac{p(\beta)}{p(Y)} \sum_{\Gamma_1} \cdots \sum_{\Gamma_n} \prod_i [p(Y_i | \Gamma_i, \beta) p(\Gamma_i)]$$

Breaking one factor out of the product gives

$$p(\beta | Y) = \frac{p(\beta)}{p(Y)} \sum_{\Gamma_1} \cdots \sum_{\Gamma_{n-1}} \prod_{i=1}^{n-1} [p(Y_i | \Gamma_i, \beta) p(\Gamma_i)] \sum_{\Gamma_n} p(Y_n | \Gamma_n, \beta) p(\Gamma_n)$$

Continuing in similar fashion yields

$$p(\beta | Y) = \frac{p(\beta)}{p(Y)} \prod_i \left[\sum_{\Gamma_i} p(Y_i | \Gamma_i, \beta) p(\Gamma_i) \right]$$

This may be written as

$$p(\beta | Y) = \frac{p(\beta)}{p(Y)} \prod_i p(Y_i | \beta) \quad (10)$$

since

$$p(Y_i | \beta) = \sum_{\Gamma_i} p(Y_i | \Gamma_i, \beta) p(\Gamma_i) \quad (11)$$

Splitting the Γ_i sum into its cases,

$$p(Y_i | \beta) = p(Y_i | \Gamma_i = \perp, \beta) p(\Gamma_i = \perp) + \sum_{M_j} p(Y_i | \Gamma_i = M_j, \beta) p(\Gamma_i = M_j)$$

Substituting the densities assumed in the model of Section 5 in Equations 6 and 2 yields

$$p(Y_i | \beta) = \frac{1}{W_1 \cdots W_v} B_i + \sum_{M_j} G_{\psi_{ij}}(Y_i - \mathcal{P}(M_j, \beta)) \frac{1 - B_i}{m} \quad (12)$$

Installing this into Equation 10 leads to

$$p(\beta | Y) = \frac{B_1 B_2 \cdots B_n}{(W_1 W_2 \cdots W_v)^n} \frac{p(\beta)}{p(Y)} \times \prod_i \left[1 + \sum_{M_j} \frac{W_1 \cdots W_v}{m} \frac{1 - B_i}{B_i} G_{\psi_{ij}}(Y_i - \mathcal{P}(M_j, \beta)) \right]$$

The objective function for posterior marginal pose estimation may be defined as the scaled logarithm of the posterior marginal probability of the pose, as follows,

$$L(\beta) \equiv \ln \left[\frac{p(\beta | Y)}{K} \right],$$

where K is a constant that has the following definition:

$$K = \frac{B_1 B_2 \cdots B_n}{(W_1 W_2 \cdots W_v)^n} (2\pi)^{-\frac{n}{2}} |\psi_\beta|^{-\frac{n}{2}} \frac{1}{p(Y)}$$

K has been chosen to simplify the form of the objective function. This leads to the following expression for the

objective function (use of a normal pose prior is assumed, as in Equation 8)

$$L(\beta) = -\frac{1}{2}(\beta - \beta_0)^T \psi_{\beta}^{-1}(\beta - \beta_0) + \sum_i \ln \left[1 + \sum_{M_j} \frac{W_1 \cdots W_v}{m} \frac{1 - B_i}{B_i} G_{\psi_{ij}}(Y_i - \mathcal{P}(M_j, \beta)) \right] \quad (13)$$

This objective function for evaluating pose hypotheses is a smooth function of the pose. Methods of continuous optimization may be used to search for local maxima, although starting values are an issue.

The first term in the PMPE objective function (Equation 13) is due to the pose prior. It is a quadratic penalty for deviations from the nominal pose. The second term essentially measures the degree of alignment of the object model with the image. It is a sum taken over image features of a smooth non-linear function that peaks up positively when the pose brings object features into alignment with the image feature. The logarithmic term will be near zero if there are no model features close to the image feature in question.

When the MRF correspondence model mentioned in Section 2.1 is used in PMPE, a simple closed form expression for the estimator no longer obtains. However, a tractable dynamic programming method for evaluating the resulting PMPE objective function at a specific pose is described in [11].

7 Expectation - Maximization Algorithm

The Expectation - Maximization (EM) algorithm was introduced in its general form by Dempster, Rubin and Laird in 1978 [16]. It is often useful for computing estimates in domains having two sample spaces, where the events in one are unions over events in the other. This situation holds among the sample spaces of posterior marginal pose estimation (PMPE) and MAP model matching. In the original paper, the wide generality of the EM algorithm is discussed, along with several previous appearances in special cases, and convergence results are described.

In this section, a specific form of the EM algorithm is described for use with PMPE. It is used for hypothesis refinement in the recognition experiments that are described in Section 8. A linear model is assumed for feature projection.

In PMPE, the pose of an object, β , is estimated by maximizing its posterior probability, given an image.

$$\hat{\beta} = \arg \max_{\beta} p(\beta | Y) .$$

A necessary condition for the maximum is that the gradient of the posterior probability with respect to the pose be zero, or equivalently, that the gradient of the logarithm of the posterior probability be zero:

$$0 = \nabla_{\beta} \ln p(\hat{\beta} | Y) . \quad (14)$$

Imposing the condition of Equation 14 on the posterior probability of the pose of an object, given an image, of Equation 10 yields the following,

$$0 = \frac{\nabla_{\beta} p(\hat{\beta})}{p(\hat{\beta})} + \sum_i \frac{\nabla_{\beta} p(Y_i | \hat{\beta})}{p(Y_i | \hat{\beta})} \quad (15)$$

Using the linear projection model, Equation 12 may be re-written as

$$p(Y_i | \beta) = \frac{B_i}{W_1 W_2 \cdots W_v} + \frac{1 - B_i}{m} \sum_j G_{\psi_{ij}}(Y_i - M_j \beta)$$

The zero gradient condition of Equation 15 may now be expressed as follows,

$$0 = \frac{\nabla_{\beta} p(\hat{\beta})}{p(\hat{\beta})} + \sum_i \frac{\frac{1-B_i}{m} \sum_j \nabla_{\beta} G_{\psi_{ij}}(Y_i - M_j \hat{\beta})}{\frac{B_i}{W_1 W_2 \cdots W_v} + \frac{1-B_i}{m} \sum_j G_{\psi_{ij}}(Y_i - M_j \hat{\beta})}$$

Using the normal pose prior, and differentiating the normal densities yields

$$0 = \psi_{\beta}^{-1}(\hat{\beta} - \beta_0) + \sum_i \frac{\frac{1-B_i}{m} \sum_j G_{\psi_{ij}}(Y_i - M_j \hat{\beta}) M_j^T \psi_{ij}^{-1}(Y_i - M_j \hat{\beta})}{\frac{B_i}{W_1 W_2 \cdots W_v} + \frac{1-B_i}{m} \sum_j G_{\psi_{ij}}(Y_i - M_j \hat{\beta})}$$

Finally, the zero gradient condition may be expressed compactly as follows,

$$0 = \psi_{\beta}^{-1}(\hat{\beta} - \beta_0) + \sum_{ij} W_{ij} M_j^T \psi_{ij}^{-1}(Y_i - M_j \hat{\beta}) , \quad (16)$$

with the following definition:

$$W_{ij} = \frac{G_{\psi_{ij}}(Y_i - M_j \hat{\beta})}{\frac{B_i}{1-B_i} \frac{1}{W_1 W_2 \cdots W_v} + \sum_j G_{\psi_{ij}}(Y_i - M_j \hat{\beta})} \quad (17)$$

Equation 16 has the appearance of being a linear equation for the pose estimate $\hat{\beta}$ that satisfies the zero gradient condition for being a maximum. Unfortunately, it is not a linear equation, because W_{ij} (the "weights") are not constants, they are functions of $\hat{\beta}$. To find solutions to Equation 16, the EM algorithm iterates the following two steps:

- Treating the weights, W_{ij} as constants, solve Equation 16 as a linear equation for a new pose estimate $\hat{\beta}$. This is referred to as the M step.
- Using the most recent pose estimate $\hat{\beta}$, re-evaluate the weights, W_{ij} , according to Equation 17. This is referred to as the E step.

The M step is so named because, in the exposition of the algorithm in [16], it corresponds to a maximum likelihood estimate. As discussed there, the algorithm is also amenable to use in MAP formulations (like this one). Here the M step corresponds to a MAP estimate of the pose, given that the current estimate of the weights is correct.

The E step is so named because calculating the W_{ij} corresponds to taking the expectation of some random variables, given the image data, and that the most recent pose estimate is correct. These random variables have value 1 if the i 'th image feature corresponds to the j 'th object feature, and 0 otherwise. Thus, after the iteration converges, the weights provide a continuous-valued estimate of the correspondences, that varies between 0 and 1.

It seems somewhat ironic that, having abandoned the correspondences as being part of the hypothesis in the formulation of PMPE, a good estimate of them has reappeared as a byproduct of a method for search in pose space. This estimate, the posterior expectation, is the minimum variance estimator.

Being essentially a local method of non-linear optimization, the EM algorithm needs good starting values in order to converge to the right local maximum. It may be started on either step. If it is started on the E step, an initial pose estimate is required. When started on the M step, an initial set of weights is needed.

An initial set of weights can be obtained from a partial hypothesis of correspondences in a simple manner. The weights associated with each set of corresponding features in the hypothesis are set to 1, the rest to 0. Indexing methods are one source of such hypotheses.

8 Range Feature Recognition Experiment

In this section, an experiment is described where the EM algorithm is used in a coarse - fine scheme to estimate the pose of a vehicle appearing in synthetic range images, without the need for feature correspondence information. The region of convergence of the coarse - fine algorithm is explored. The object has four degrees of freedom - translation, rotation, and scaling in the plane. Similar experiments, with full freedom of motion in 3D are described in [11].

8.1 Preparation of Features

The preparation of the features used in the experiment is summarized in Figure 4. The features were oriented-range features, as described in Section 4.1. Two sets of features were prepared, the "model features", and the "image features".

The object model features were derived from a synthetic range image of an M35 truck that was created using the ray tracing program associated with the BRL CAD Package [17]. The ray tracer was modified to produce range images instead of shaded images. The synthetic range image appears in the first image of Figure 5.

In order to simulate a laser radar, the synthetic range image described above was corrupted with simulated laser radar sensor noise, using a sensor noise model that is described by Shapiro, Reinhold, and Park [18]. In this noise model, measured ranges are either valid or anomalous. Valid measurements are normally distributed, and anomalous measurements are uniformly distributed. The corrupted range image appears as the

second image in Figure 5. To simulate post sensor processing, the corrupted image was "restored" via a statistical restoration method of Menon and Wells [19]. The restored range image appears as the third image of Figure 5.

Oriented range features, as described in Section 4.1, were extracted from the synthetic range image, for use as model features - and from the restored range image, these are called the noisy features. The features were extracted from the range images in the following manner. Range discontinuities were located by thresholding neighboring pixels, yielding range discontinuity curves. These curves were then segmented into approximately 20-pixel-long segments via a process of line segment approximation. The line segments (each representing a fragment of a range discontinuity curve) were then converted into oriented range features in the following manner. The X and Y coordinates of the feature were obtained from the mean of the pixel coordinates. The normal vector to the pixels was gotten via least-squares line fitting. The range to the feature was estimated by taking the mean of the pixel ranges on the near side of the discontinuity. This information was packaged into an oriented-range feature, as described in Section 4.1. The model features are shown in the first image of Figure 6. Each line segment represents one oriented-range feature, the ticks on the segments indicate the near side of the range discontinuity. There are 113 such object features.

The noisy features, derived from the restored range image, appear in the second image of Figure 6. There are 62 noisy features. Some features have been lost due to the corruption and restoration of the range image. The set of image features was prepared from the noisy features by randomly deleting half of the features, transforming the survivors according to a test pose, and adding sufficient randomly generated features so that $\frac{1}{8}$ of the features are due to the object. The 248 image features appear in the third image of Figure 6.

8.2 Coarse-Fine Method

A coarse-fine search method was used for finding maxima of the pose-space objective function. Two levels of smoothing the objective function were used. Peaks, initially located at the coarsest scale, are used as starting values for a search at the next (less smooth) scale. Finally, results of the second level search are used as the initial values for search in the un-smoothed objective function. This coarse-fine method combines the accuracy of the un-smoothed objective function with the larger region of convergence of the smoothed objective function.

The objective function was smoothed by replacing the stationary feature covariance matrix $\hat{\psi}$ in the following manner:

$$\hat{\psi} \leftarrow \hat{\psi} + \psi_s$$

The effect of the smoothing matrix ψ_s is to increase the spatial scale of the covariance matrices that appear in the objective function. The smoothing matrices used in the experiment were as follows,

$$\text{DIAG}((2.0)^2, (2.0)^2, (.01)^2, (.01)^2)$$

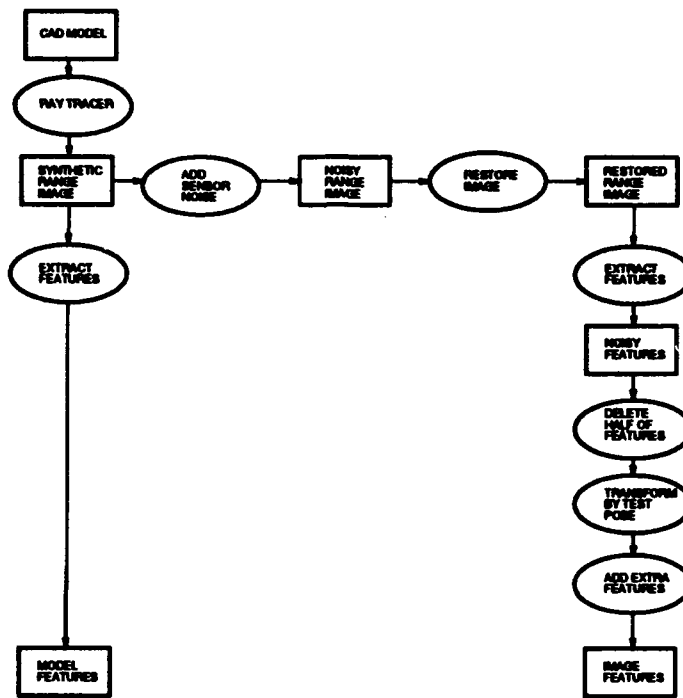


Figure 4: Preparation of Features

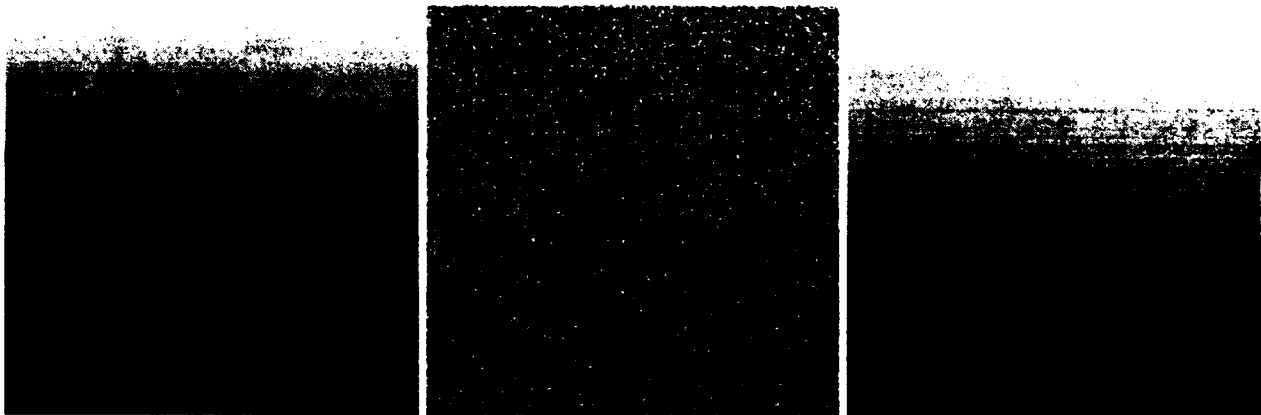


Figure 5: Synthetic Range Image, Noisy Range Image, and Restored Range Image

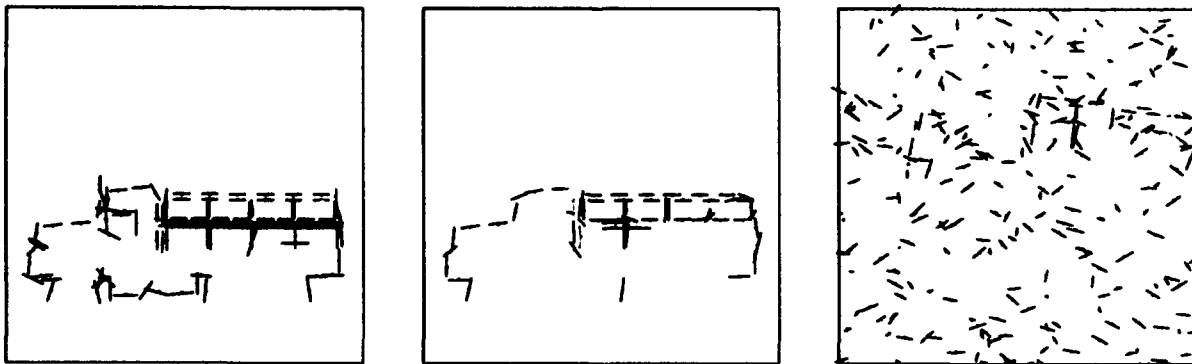


Figure 6: Model Features, Noisy Features, and Image Features

and

$$\text{DIAG}((5.0)^2, (5.0)^2, (.025)^2, (.025)^2)$$

where $\text{DIAG}(\cdot)$ constructs diagonal matrices from its arguments. These smoothing matrices were determined empirically.

8.3 Results

Figure 7 illustrates the approximate region of convergence of the coarse-fine search with the EM algorithm described above, when used with the image and model features described in Section 8.1. Eight poses are displayed with the truck shown with light lines – the true pose of the truck is shown for reference with heavy lines. These eight poses are displacements from the true pose in both directions along the four coordinate axes of the pose space. They represent, approximately, the region of convergence of the coarse-fine method – the algorithm converges from these poses, but not from much farther away from the true pose.

An image displaying a sequence of poses from an EM iteration at the coarsest scale appears in Figure 8. The algorithm converged in 21 iterations.

9 Related Work

Green [20] and Shapiro and Green [21] describe a theory of maximum-likelihood laser radar range profiling. The research focuses on statistically optimal detectors and recognizers. The single pixel statistics are described by a mixture of uniform and normal components. Range profiling is implemented using the EM algorithm. Under some circumstances, least squares provides an adequate starting value. A continuation-style variant is described, where a range accuracy parameter is varied between EM convergences from a coarse value to its true value.

Cass [22] describes an approach to visual object recognition that searches in pose space for maximal alignments under the bounded-error model. The pose-space objective function used there is piecewise constant, and is thus not amenable to continuous search methods. The search is based on a geometric formulation of the constraints on feasible transformations.

There are some connections between PMPE and standard methods of robust pose estimation, like that described by Haralick [23]. Both can provide robust estimates of the pose of an object, based on an observed image. The main difference is that the standard methods require specification of the feature correspondences, while PMPE does not – by considering all possible correspondences. PMPE requires a starting value for the pose (as do standard methods of robust pose estimation that use non-convex objective functions).

As mentioned above, Yuille, Geiger and Bülthoff [1] discussed computing disparities in a statistical theory of stereo where a marginal is computed over matches. Yuille extends this technique [24] to other domains of vision and neural networks, among them winner-take-all networks, stereo, long-range motion, the traveling salesman problem, deformable template matching, learning, content addressable memories, and models of brain development. In addition to computing marginals over discrete fields, the Gibbs probability distribution is used.

This facilitates continuation-style optimization methods by variation of the temperature parameter. There are some similarities between this approach and using coarse-fine with the PMPE objective function.

There is a similarity between posterior marginal pose estimation and Hough transform (HT) methods. Roughly speaking, HT methods evaluate parameters by accumulating votes in a discrete parameter space, based on observed features. (See the survey paper by Illingworth and Kittler [25] for a discussion of Hough methods.)

In a recognition application, as described here, the HT method would evaluate a discrete pose by counting the number of feature pairings that are exactly consistent somewhere within the cell of pose space. As stated, the HT method has difficulties with noisy features. This is usually addressed by counting feature pairings that are exactly consistent somewhere nearby the cell in pose space.

The utility of the HT as a stand-alone method for recognition in the presence of noise is a topic of some controversy. This is discussed by Grimson in [26], pp. 220. Perhaps this is due to an unsuitable noise model implicit in the Hough Transform.

PMPE evaluates a pose by accumulating the logarithm of posterior marginal probability of the pose over image features.

The connection between HT methods and parameter evaluation via the logarithm of posterior probability has been described by Stephens [27]. Stephens proposes to call the posterior probability of parameters given image observations "The Probabilistic Hough Transform." He provided an example of estimating line parameters where image feature point probability densities were described as having uniform and normal components. He also states that the method has been used to track 3D objects, referring to his thesis [28] for definition of the method used.

Lipson [29] describes a system that does refinement of alignments under Linear Combination of Views. The method iterates solving linear systems. It differs by matching model features to the nearest image feature under the current pose hypothesis, while the method described here entertains matches to all of the image features, weighted by their probability.

10 Acknowledgments

I thank my thesis advisor, W. E. L. Grimson, for his patience and guidance in the course of this research. I am grateful to Patrick Winston for making MIT's AI Lab the unique place that it is. During several summers this research was supported at MIT Lincoln Laboratory, group 53. Group leader Al Gschwendtner provided a stimulating environment for recognition research in laser radar imagery. There, I appreciate the help Steve Rak and Murali Menon offered me in preparing the images used in the experiments of Section 8.

References

- [1] A. Yuille, D. Geiger, and H. Bülthoff. Stereo In-

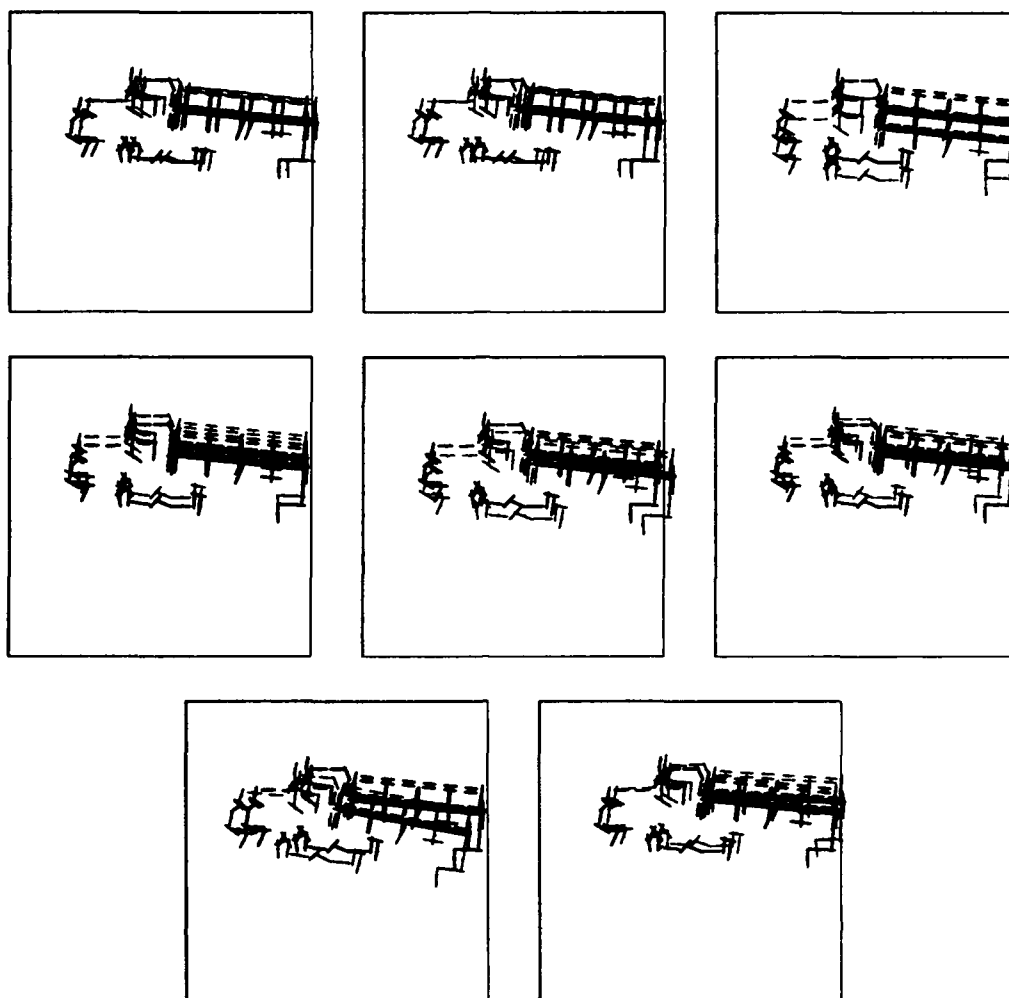


Figure 7: Eight Convergent Starting Poses

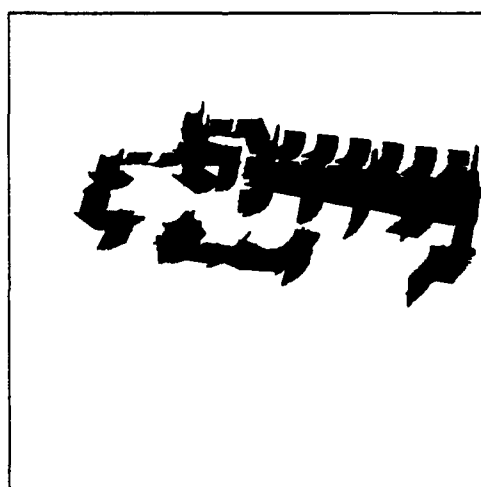


Figure 8: EM Iteration Poses

- tegration, Mean Field Theory and Psychophysics. In *Computer Vision - ECCV 90*, pages 73-82. Springer Verlag, 1990.
- [2] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions PAMI*, PAMI-6(6):721-741, November 1984.
- [3] S.T. Barnard. Stereo Matching by Hierarchical, Microcanonical Annealing. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 832-835, August 1987.
- [4] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, 1987.
- [5] J. Beveridge, R. Weiss, and E. Riseman. Optimization of 2-Dimensional Model Matching. In *Proceedings: Image Understanding Workshop*, pages 815-830. Morgan Kaufmann, 1989.
- [6] D.P. Huttenlocher and S. Ullman. Recognizing Solid Objects by Alignment. In *Proceedings: Image Understanding Workshop*, pages 1114-1124. Morgan Kaufmann, April 1988.
- [7] Y. Lamdan and H.J. Wolfson. Geometric Hashing: A General and Efficient Model-Based Recognition Scheme. In *Second Int. Conf. Comp. Vision*, 1988.
- [8] D.T. Clemens and D.W. Jacobs. Model Group Indexing for Recognition. In *Symposium on Advances in Intelligent Systems*. SPIE, 1990.
- [9] W.E.L. Grimson and T. Lozano-Pérez. Localizing Overlapping Parts by Searching the Interpretation Tree. *IEEE Transactions PAMI*, PAMI-9(4):469-482, July 1987.
- [10] T.M. Breuel. *Geometric Aspects of Visual Object Recognition*. PhD thesis, MIT Department of Brain and Cognitive Sciences, 1992.
- [11] W.M. Wells III. *Statistical Object Recognition*. PhD thesis, MIT Department Electrical Engineering and Computer Science, Cambridge, Mass., 1992. MIT AI Laboratory TR 1398.
- [12] W.M. Wells III. MAP Model Matching. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition*, pages 486-492, Lahaina, Maui, Hawaii, June 1991. IEEE.
- [13] N. Ayache and O.D. Faugeras. HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects. *IEEE Transactions PAMI*, PAMI-8(1):44-54, January 1986.
- [14] W.M. Wells III. A Statistical Approach to Model Matching. In *Symposium on Advances in Intelligent Systems*. SPIE, 1990.
- [15] W.M. Wells III. Posterior Marginal Pose Estimation. In *Proceedings: Image Understanding Workshop*, pages 745 - 751. Morgan Kaufmann, January 1992.
- [16] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. Roy. Statist. Soc.*, 39:1 - 38, 1977.
- [17] P. Dykstra and M.J. Muuss. The BRL CAD Package: an Overview. In *Proceedings of the Fourth USENIX Computer Graphics Workshop*, pages 73-80, Cambridge MA, 1987.
- [18] J.H. Shapiro, R.W. Reinhold, and D. Park. Performance Analyses for Peak-Detecting Laser Radars. *Proceedings of the SPIE*, 663:38-56, 1986.
- [19] M. Menon and W.M. Wells III. Massively Parallel Image Restoration. In *Proceedings of the International Joint Conference on Neural Networks*, San Diego, CA., 1990. IEEE.
- [20] T.J. Green, Jr. *Three-Dimensional Object Recognition Using Laser Radar*. PhD thesis, MIT Department Electrical Engineering and Computer Science, 1992.
- [21] T.J. Green, Jr. and J.H. Shapiro. Maximum-Likelihood Laser Radar Range Profiling with the Expectation - Maximization Algorithm. *Opt. Eng.*, November 1992.
- [22] T.A. Cass. Polynomial-Time Object Recognition in the Presence of Clutter, Occlusion, and Uncertainty. In G. Sandini, editor, *Computer Vision - ECCV '92*, pages 834-851. Springer Verlag, 1992.
- [23] R.M. Haralick, H.Joo, C.N.Lee, X.Zhuang, V.G.Vaidya, and M.B.Kim. Pose Estimation from Corresponding Point Data. *IEEE Trans. on Systems Man and Cybernetics*, 19(6):1426 - 1445, December 1989.
- [24] A.L. Yuille. Generalized Deformable Models, Statistical Physics, and Matching Problems. *Neural Computation*, 2:1 - 24, 1990.
- [25] J. Illingworth and J. Kittler. A Survey of the Hough Transform. *Computer Vision, Graphics, and Image Processing*, 44:87-116, 1988.
- [26] W.E.L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints*. MIT Press, 1990.
- [27] R.S. Stephens. A Probabilistic Approach to the Hough Transform. In *Proceedings of the British Machine Vision Conference*, pages 55-60, Univ. of Oxford, September 1990.
- [28] R.S. Stephens. *The Hough Transform: A Probabilistic Approach*. PhD thesis, Cambridge University, Department of Engineering, 1990.
- [29] P. Lipson. Model Guided Correspondence. Master's thesis, MIT Department Electrical Engineering and Computer Science, 1992.

Scalable Data Parallel Geometric Hashing: Experiments on MasPar MP-1 and on Connection Machine CM-5

Ashfaq Khokhar and Viktor K. Prasanna *

(Summary of Results)

Abstract

Geometric hashing has been recently proposed as a technique for model based object recognition in occluded scenes. However, this technique is computationally demanding; a probe of the recognition phase on a serial machine can take several minutes to complete. In this paper, we present parallel algorithms and several fast parallel implementations on MasPar MP-1, a Single Instruction Multiple Data (SIMD) machine, and on the Connection Machine CM-5 operating in Single Program Multiple Data (SPMD) mode. Based on the results, we compare the merits of the above classes of architectures for vision. In earlier parallel implementations, the number of processors employed is independent of the size of the scene but depends on the size of the model database which is usually very large. We design new parallel algorithms and map them onto MP-1 and onto CM-5. These techniques significantly improve upon the number of processors employed while at the same time achieve much superior time performance. Earlier implementations claim 700 to 1300 msec for a probe of the recognition phase, assuming 200 feature points in the scene on an 8K processor CM-2. Our implementations run on a P processor machine, such that $1 \leq P \leq S$, where S is the number of feature points in the scene. Our results show that a probe of the recognition phase for a scene consisting of 1024 feature points takes less than 50 msec on a 1K processor MP-1 and it takes less than 10 msec on a 256 processor CM-5. The model database used in these implementations contains 1024 models and each model is represented by 16 feature points. The implementations developed in this paper require number of processors independent of the size of the model database and are scalable with the machine size. Results of concurrent processing of multiple probes of the recognition

phase are also reported.

1 Introduction

This paper is in continuation of our efforts in studying parallel techniques for solving high level vision problems [10; 9; 11; 12; 16]. In this paper, we present scalable parallel techniques to implement the object recognition problem using geometric hashing. Object recognition, a high level vision task, is a key step in an integrated vision system. Most model based recognition systems work by hypothesizing matches between scene features and model features, predicting new matches, and verifying or changing the hypotheses through a search process [3; 4; 5; 6; 19]. Geometric hashing [20] offers a different and more parallelizable paradigm. However, parallel techniques are needed to use geometric hashing in real time applications.

In geometric hashing, given a set of models and their features points, for each model, all possible pairs of the feature points are designated as a *basis set*. The coordinates of the features points of a model are computed relative to each member of its basis set. These coordinates are then used as indices into a hash table. The records in the hash table comprise of (*model, basis*) pairs. In the recognition phase, an arbitrary pair of feature points in the scene is chosen as basis and coordinated of the feature points in the scene are computed. The new coordinates are used to hash into the hash table and the corresponding entries of the hashed bin are accessed. The (*model, basis*) pair winning maximum number of votes is chosen as candidate for matching.

There have been two prior efforts in parallelizing the geometric hashing algorithm [1; 18]. Both implementations have been performed on SIMD hypercube based machines. These implementations are among the early efforts in using parallel techniques to solve high-level vision problems. One of the major problems in both the implementations is the requirement of large number of processors. In our results, we exploit the fact that the number of votes cast in an iteration of the recognition phase is bounded by S , the number of feature points in the scene. Therefore, no more than S locations of the hash table are accessed during the execution of the recognition algorithm. This allows us to reduce the number of processors employed to at most S , the number of fea-

*This research was supported by the Defense Advanced Research Projects Agency under contract F49620-90-C-0078, monitored by the Air Force Office of Scientific Research and in part by the Army Research Office contract number DAAL03-89-C-0038 with the University of Minnesota Army High Performance Computing Research Center. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

Address: Department of EE-Systems, EEB 244, University of Southern California, Los Angeles, CA 90089-2562, email: {ashfaq + prasanna}@halcyon.usc.edu

ture points in a scene. Previous implementations used $O(Mn^3)$ processors, i.e., the number of bins in the hash table, where M is the number of models in the database and n is the number of feature points in each model. In addition, the implementation by Bourdon and Medioni [1] suffers due to inefficiency of the routing algorithm. This inefficiency limits the scope of their implementation to a small model database. In [18], Rigoutsos and Hummel suggest to use radix sort to implement histogramming, a technique used in their implementation to count the votes for each (*model, basis*) pair. The use of radix sort in histogramming is advantageous only if the number of levels in the histogram is much less than the number of data points [15]. In case of geometric hashing, this is not true. In this paper, we present several fast parallel techniques to implement the object recognition problem using geometric hashing on SIMD and SPMD machines. We also provide implementation results of the techniques developed in this paper on the Connection Machine CM-5 operating in SPMD mode, and on MasPar MP-1, an SIMD machine.

In our implementations, we provide various partitioning, mapping and routing techniques to address the above issues. These lead to significantly less number of processors to be used, while achieving much superior time performance. Earlier implementations [1; 18] claim 700 to 1300 msec for one probe of the recognition phase, assuming a scene of 200 feature points, on an 8K processor CM-2. We provide techniques to implement the recognition phase on a P processor array, such that $1 \leq P \leq S$, where S is the number of feature points in the scene. Our results show that one probe of the recognition phase for a scene consisting of 1024 feature points takes less than 50 msec on a 1K processor MP-1 and it takes less than 10 msec on a 256 processor CM-5. The model database used in the implementations contains 1024 models and each model is represented using 16 feature points. The implementations developed in this paper require number of processors independent of the size of the model database and are scalable with the machine size. Results of concurrent processing of multiple probes of the recognition phase are also reported. Based on the implementation results, we compare the merits of classes of machines used for vision algorithms.

The organization of the paper is as follows. The geometric hashing technique for object recognition is outlined in Section 2. Section 3 discusses parallelism in geometric hashing. In Section 4, implementation details are shown and experimental results are tabulated and compared. Conclusions are presented in Section 5. This paper summarizes our work. Related results and additional details can be found in [11; 7].

2 Object Recognition Using Geometric Hashing

In a model-based recognition system, a set of objects is given and the task is to find instances of these objects in a given scene. The objects are represented as sets of geometric features, such as points or edges, and their ge-

ometric relations are encoded using a minimal set of such features. The task becomes more complex if the objects overlap in the scene and/or other occluded unfamiliar objects exist in the scene.

Many model based recognition systems are based on hypothesizing matches between scene features and model features, predicting new matches, and verifying or changing the hypotheses through a search process. Geometric hashing, introduced by Lamdan and Wolfson [20], offers a different and more parallelizable paradigm. It can be used to recognize flat objects under weak perspective. For the sake of completeness, we briefly outline the geometric hashing technique in Section 2.1. Additional details can be found in [20].

2.1 Geometric Hashing Algorithm

The algorithm consists of two procedures, preprocessing and recognition. These are shown in Figures 1 and 2 respectively.

Preprocessing:

The preprocessing procedure is executed off-line and only once. In this procedure, the model features are encoded and are stored in a hash table data structure. However, the information is stored in a highly redundant multiple-viewpoint way. Assume each model in the database has n feature points. For each ordered pair of feature points in the model chosen as basis, the coordinates of all other points in the model are computed in the orthogonal coordinate frame defined by the basis pair. Each such coordinate is quantized and is used as an entry to a hash table, where the (*model, basis*) pair, at which the coordinate was obtained, is recorded. The complexity of this preprocessing procedure is $O(n^3)$ for each model, hence $O(Mn^3)$ for M models.

Recognition:

In the recognition procedure, a scene consisting of S feature points is given as input. An arbitrary ordered pair of feature points in the scene is chosen. Taking this pair as a basis, the coordinates of the remaining feature points are computed. Each such coordinate is used as a key to enter the hash table (constructed in the preprocessing phase), and for every recorded (*model, basis*) pair at the corresponding location, a vote is collected for that pair. The pair winning the maximum number of votes is taken as a matching candidate. The execution of the recognition phase corresponding to one basis pair is termed as a *probe*. Finally, edges of the matching candidate model are verified against the scene edges. If no (*model, basis*) pair scores high enough, another basis from the scene feature points is chosen and probe is executed. Therefore, the worst case time complexity of the recognition procedure is $O(S^3)$. However, if some classification for choosing a basis from the scene is available, the complexity can be reduced to $O(S)$ [13].

The time taken per probe depends on the hash function employed. The vision community has experimented

Preprocessing()

```
for each model  $i$  such that  $1 \leq i \leq M$  do
  Extract  $n$  feature points from the model;
  for  $j = 1$  to  $n$ 
    for  $k = 1$  to  $n$ 
      - Compute the coordinates of all other features
        points in the model by taking this pair as basis.
      - Quantize each of the above computed coordinates
        and use it as a key to enter into a hash table
        where the pair (model, basis), i.e., ( $i, jk$ ),
        is recorded.
    next  $k$ 
  next  $j$ 
next  $i$ 
end
```

Figure 1: A sequential procedure to construct a hash table consisting of (*model*, *basis*) pairs.

Recognition()

1. Extract S feature points from the scene;
 2. *Selection*:
Select a pair of feature points as basis
 3. *Probe*:
 - a. Compute the coordinates of all other features points in the scene relative to the selected basis.
 - b. Quantize each of the above computed coordinates and use it as a key to access the hash table containing the entries of the (*model*, *basis*) pairs.
 - c. Vote for the entries in the hash table.
 - d. Select the (*model*, *basis*) pair with the maximum votes as the matched model in the scene.
 4. *Verification*:
Verify the candidate model edges against the scene edges.
 5. If the model wins the verification process, remove the corresponding feature points from the scene.
 6. Repeat steps 2, 3, 4, and 5 until some specified condition.
- end

Figure 2: Outline of the steps in sequential recognition.

with various hash functions and hash functions distributing the feature points uniformly into the hash table are known [18]. We will be using these hash functions in our implementations. Assuming that S feature points of the input scene leads to $O(S)$ total number of votes, the voting process in a probe of the recognition phase can be implemented in $O(S \log S)$ time using sorting. Other parts of the computation are time consuming, even though they do not contribute to the time complexity. Note that the total number of (*model*, *basis*) pairs is $O(Mn^2)$. The voting time can be reduced to $O(S + Mn^2)$ by employing $O(Mn^2)$ boxes to collect the votes. Through out this paper, we assume $S \ll Mn^2$.

3 Scalable Data Parallel Geometric Hashing

In this section, we present parallel techniques to implement the recognition phase on a P processor machine. Algorithms presented in this section are implemented on Connection Machine CM-5 operating in SPMD mode and on MasPar MP-1, an SIMD array. In an SIMD machine, each processor executes a stream of instructions in a lock-step mode on the data available in its local memory. The instructions are broadcast by the control unit. The SPMD mode of execution combines the characteristics of SIMD and MIMD modes. In this mode, the control processor broadcasts a section of the data parallel program to the processing nodes, rather than broadcasting an instruction at a time (as in a typical SIMD machine). At the start of the execution of a program, the complete program is sent to all the nodes with pseudo synchronization instructions embedded in the code. Each node executes the program independent of others until an embedded synchronization instruction is reached. It resumes the execution of the program only after all the nodes have reached the synchronization barrier. The control unit assists in enforcing the synchronization barriers embedded in the program. This operation mode is also referred to as *synchronized MIMD* mode [2].

We will not elaborate on parallelizing the preprocessing phase, since it is a one time process and can be carried out off-line. However, details of that procedure can be found in [7]. The size of the model database (the number of hash bins) is $O(Mn^3)$.

3.1 Partitioned Implementation of the Recognition Procedure

We use P processors such that $1 \leq P \leq S$, where S is the number of feature points in a scene. Each Processing Element (PE) in the array is assumed to have $O(\frac{Mn^3}{P})$ memory.

The input is a scene consisting of S feature points. In the recognition phase, possible occurrence of the models (stored in the database) in the scene is checked. The models are available in a hash table created during preprocessing. All the models are allowed to go under rigid and/or similarity transformations. An arbitrary ordered pair of feature points in the scene is chosen. Taking this pair as a basis, a probe of the model data base is performed. The main steps of a parallel algorithm to process a single probe of the recognition phase are given in Figure 3.

As we are using less number of processors than the size of the hash table, each PE will have several hash table bins stored in its local memory. Two issues arise during the execution of the procedure *ParallelProbe()*.

1. More than one feature point in the scene may cast their votes to the same location in the hash table, resulting in a contention for a single memory location in a PE (see Figure 4).
2. More than one feature point in the scene may cast their votes to different bins stored in a PE, resulting in a congestion at a PE (see Figure 5).

Parallel_Probe(S, P)

/* S is the number of features in the input scene and P is the number of processors. Initially each PE is assumed to have S/P distinct scene feature points stored in a local array $FP[]$. */

- Choose an arbitrary pair of feature points in the scene as a basis and broadcast it to all the PEs.
 - Compute_Keys()
 - Vote()
 - Compute_Winner()
- end

Figure 3: A parallel procedure to process a probe of the recognition phase.

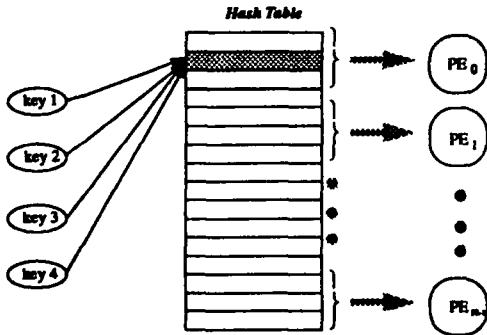


Figure 4: Contention for a single hash bin.

The worst case in both the cases will be $O(S)$ contention and congestion. Such a scenario can lead to no speedup at all. On the other hand, other researchers have used large number of processors to avoid congestion and contention problems. However, in their implementations the processor utilization is extremely low. Also, such solutions result in enormous communication overheads in performing global operations, such as global max and histogramming, as evident in the implementations proposed in [1; 18]. In the following, we address these issues and present efficient mapping and routing techniques to resolve the contention and congestion problems arising in performing a probe, while using a small number of processors.

In order to eliminate the memory contention problem in the array, we introduce a *Merge_Key()* procedure shown in Figure 6. This procedure sorts the hash table keys corresponding to the input scene. The keys having the same key value reside in a block of PEs and in each block the least indexed PE holds the *leader key*. The leader key has the sum of the number of elements in its block. During the voting process, each leader key accesses the PE holding the corresponding location of the hash table and casts a vote on behalf of all the keys in its block, i.e. if there are m elements in the block, m votes will be registered for the corresponding location in the hash table. This reduces the number of accesses to the hash table stored in a PE and thus reduces the traffic

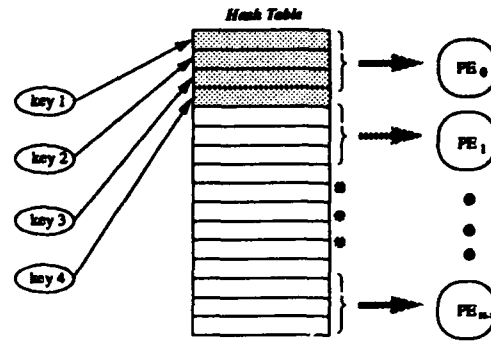


Figure 5: Congestion at a PE while accessing different hash bins stored in a PE.

over the network.

Similarly, in order to address the processor *congestion* problem, we propose to store multiple copies of the hash table in the array. In the worst case, a copy of the hash table can be assigned to each sub-array of suitable size. This increases the size of the memory required within each processor. Each PE restricts its search for the hash table bins within the corresponding subarray. This solution localizes the congestion problem to the sub arrays.

Merge_Keys(INPUT)

- Sort(INPUT)
 - In parallel, each PE_i , $0 \leq i \leq P-1$
for each distinct key j , $0 \leq j \leq S/P-1$.
Identify the *leader key* and mark it in the array $INPUT[j]$.
 - In parallel, each PE_i , $0 \leq i \leq P-1$
for each *leader key*,
Count the number of keys same as the *leader key* and store it in $OUTPUT[]$.
- end

Figure 6: A parallel procedure to merge quantized coordinates of feature points of the scene.

In the following analysis, we ignore the initialization costs, such as loading the scene points to the processor array, loading hash table locations to the processor array, and initialization of memory locations used inside each PE. These assumptions are also made in the previous implementations reported in [1; 18].

For asymptotic time analysis, we employ a model of CM-5 operating in SIMD mode of operation. The *fat tree* [14] is the underlying interconnection network of CM-5. We assume an SIMD mesh array, shown in Figure 8, for asymptotic time analysis on MP-1.

Theorem 1 Given a *fat tree* architecture consisting of P leaf nodes, one probe of the recognition phase can be processed in $O(\frac{S}{P} \log S)$ time on a scene consisting of S feature points, where $\log^2 P \leq \log S$. \square

The restriction on P can be relaxed if sufficient number of copies of the hash table is available. Based on the

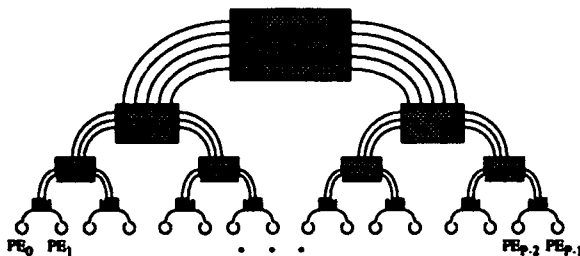


Figure 7: A fat tree model.

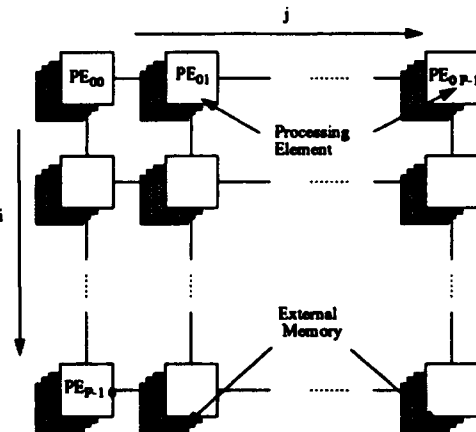


Figure 8: A mesh array model.

above theorem, the algorithm for the recognition phase is processor time optimal and scales linearly with P for $1 \leq P \leq S^{1/2}$.

Theorem 2 Given a mesh array of $\sqrt{P} \times \sqrt{P}$ processors, such that $P = S$, one probe of the recognition phase can be processed in $O(\sqrt{P})$ time on a scene consisting of S feature points. \square .

Detailed analysis of the running time on these architectures can be found in [7].

4 Implementation Details and Experimental Results

In this section, first, we describe the underlying models of the machines used in our implementations and then present data mapping and partitioning strategies employed on these machines.

4.1 The Connection Machine CM-5

A Connection Machine Model CM-5 system contains between 32 and 16,348 processing nodes. Each node is a 32 MHz SPARC processor with upto 32 Mbytes of local memory. A 64 bit floating point vector processing unit is optional with each node. Each processing node is a general purpose computer that can fetch and interpret its own instruction stream. System administration tasks and serial user tasks are performed by control processors. Input and output is performed via high-bandwidth I/O interfaces.

4.2 The MasPar MP-1

The MP-1 is a massively parallel SIMD computer system with upto 16K Processing Elements. The system consists of a high performance Unix Workstation as Front End (FE) and a Data Parallel Unit (DPU). The DPU consists of PEs, each with upto 64 Kbytes of memory and 192 bytes of register space. All PEs execute instructions broadcast by an Array Control Unit (ACU) in lock step. PEs have indirect addressing capability and can be selectively disabled.

The machines used in the implementations differ w.r.t. mode of operation, processing speed of the PEs, and interprocessor communication bandwidth.

In CM-5, a coarse grain SPMD machine, each processing element (PE) is a powerful SPARC processor. The high computing power of each PE and relatively expensive communication among processors motivates to partition the data such that the algorithms exhibit less communication among PEs at the cost of redundant computation within each PE. However, a balance between these two is needed to attain speed-ups.

In MP-1, a fine grain massively parallel SIMD machine, each PE is a 4 bit processor. Interprocessor communication is supported through two communication networks, 1) Xnet for regular communication and 2) router for random communication. As in the geometric hashing algorithm the communication pattern is irregular, the router network provides superior performance over the Xnet [17]. It is also experienced that the ratio of unit-floating-point-computation time over unit-communication time (thru Xnet or thru contention free router) is approximately 1 [8]. It suggests to carefully partition the data such that both the computation and communication capabilities of the architecture are fully utilized.

4.3 Partitioning and Mapping

Three procedures, *Compute_Keys()*, *Vote()*, and *Compute_Winner()* shown in Figures 9, 10, and 11 respectively, correspond to the steps described in the *Parallel_Probe()* procedure (i.e. Figure 3). The *Compute_Keys()* procedure computes the transformed coordinates of the scene points and quantizes them according to a hash function $f()$. The transformed and quantized coordinates are stored in *NEWFP[]*. We use the same hash function as in [18]. This hash function distributes the data uniformly over all the hash bins. The transformed coordinates are then used as *keys* to access the data in the hash table. The *Vote()* procedure routes the keys to their corresponding hash locations stored in *PE_{g(key)}*. The function $g()$ defines the mapping of the hash table entries onto the processor array. The locations in the hash table accessed during voting are stored in *CANDID[]* array. This array is used in computing the final winner. The size of this array is much smaller than the size of the hash table stored in each PE.

Next, the *Compute_Winner()* procedure determines the model-basis pair having the maximum number of votes. The winning pair is then sent to the control processor to perform the final verification.

Based on the above described subtasks of the *Parallel_Probe()*

```

Compute.Keys(FP, P)
  In parallel for all  $PE_i, 0 \leq i \leq P-1$ , do
    for  $r = 0$  to  $S/P - 1$ 
      - Compute the transformed coordinate of the
        feature point  $FP[r]$  relative to the basis
        and store it in  $NEWFP[r]$ .
      - Quantize  $NEWFP[r]$  using hash function  $f()$ .
    next  $r$ 
  Parallel-end
end

```

Figure 9: A parallel procedure to compute the coordinates of feature points of the scene.

```

Vote(FPNEW, P)
  In parallel, each  $PE_i, 0 \leq i \leq P-1$ , do
     $k := 0$ ;
    for  $j = 0$  to  $S/P - 1$ 
      - Send a vote (additive write) to processor and
        location specified by  $g(NEWFP[j])$ .
      - If a vote is received, copy the contents of the
        corresponding hash-table entry in the array
         $CANDID[k++]$ .
    next  $j$ 
  Parallel-end
end

```

Figure 10: A parallel procedure to vote for the possible presence of a model in the scene.

lelProbe() procedure, several data mapping and partitioning strategies are developed, which affect the overall execution time of the recognition phase.

In the following we present four algorithms, which we have experimented with. These algorithms differ with respect to partitioning and mapping of the hash table onto the processor array. Various strategies are employed to take into account practical considerations, such as available memory in each PE, processor speed, and I/O speed of the machines. In Algorithm A, and Algorithm B, we assume that each processor is assigned $\frac{Mn^2}{P}$ distinct hash table locations. In Algorithm C, each sub-array of processors is assigned a complete copy of the hash table. Each processor in a sub-array of size, s^2 , where $1 \leq s \leq \sqrt{P}$, has $\frac{Mn^2}{s^2}$ distinct entries of the hash table.

The case of large number of processors is considered next. Algorithm D performs concurrent processing of multiple probes of the the recognition phase. The array is divided into disjoint sets of S processors. Each set of PEs processes a probe using a basis (a different basis for each set).

Algorithm A:

- Execute the *Comp.Keys()* procedure serially in the control processor and broadcast the encoded points (keys) to each processor in the processor array.
- Each processor scans through all the keys and accumulates votes for the keys

Compute.Winner()

/* Each PE is assigned $\frac{Mn^2}{P}$ distinct model-basis pairs to compute the number of votes cast to them. */

In parallel, in each $PE_i, 0 \leq i \leq P-1$
Send every element of the $CANDID[]$ array to the PE assigned for computing the total number of votes for that element.

In parallel, in each $PE_i, 0 \leq i \leq P-1$
Count the total number of votes for each distinct (*model, basis*) pair received and store it in $VCOUNT[model, basis]$.

In parallel, in each $PE_i, 0 \leq i \leq P-1$,
Compute the local maximum of $VCOUNT[]$ array and store it in *local_max*.

Compute the maximum of *local_max* over the entire processor array.

/* The (*model, basis*) pair with maximum number of votes is the matched model in the scene. */
end

Figure 11: A parallel procedure to compute the winning (*model, basis*) pair.

which correspond to hash table locations stored in its local memory.

- Executes the *Compute.Winner()* procedure.

Algorithm B:

- The control processor broadcasts S/P scene points to each processor along with a basis pair.
- Execute the *Comp.Keys()* procedure
- Execute the *Sort.Keys()* procedure.
- Execute the *Vote()* procedure
- Execute the *Compute.Winner()* procedure.

Algorithm C:

In this algorithm, we assume multiple copies of the hash table stored in the processor array.

- The control processor broadcasts S/P scene points to each processor along with a basis pair.
- Execute the *Comp.Keys()* procedure
- Execute the *Sort.Keys()* procedure.
- Execute the *Vote()* procedure such that the data search is bounded within its sub-array
- Execute the *Compute.Winner()* procedure.

Algorithm D:

In this algorithm, we assume that the number of PEs is larger than the number of feature points in the scene.

- The control processor broadcasts S/P scene points to each PE.
- The control processor broadcasts a basis pair to all PEs in each subarray of size S .

- Execute the *Comp_Keys()* procedure.
- Execute the *Sort_Keys()* procedure.
- Execute the *Vote()* procedure
- Execute the *Compute_Winner()* procedure.

4.4 Experiments and Summary of Performance Results

We have used a synthesized model database, containing 1024 models, each model consisting of 16 randomly generated points. These points are generated according to a Gaussian distribution of zero mean and unit standard deviation. The models are allowed to undergo only rigid transformation. However, results from other transformations do not affect the performance of the parallel algorithm. Similarly, scene points are synthesized using normal distribution. We apply the equalization techniques, given in [18], to the transformed coordinates, i.e., for each of the transformed point (u, v) , following hash function is applied.

$$f(u, v) = (1 - e^{-\frac{u^2 + v^2}{2\sigma^2}}) \cdot \text{atan2}(v, u)$$

The above hash function uniformly distributes the data over the hash space such that the average hash bin length is constant. We assume a data base of 1024 models with 16 points in each model. This gives a hash table size of 4M entries. Each entry may consist of several (model, basis) pairs. We have experimented on various data granularities in the hash table comprising of average bin lengths of 1, 4, 8, 16 and 32. These granularities can be chosen according to the local memory available within each PE. We have executed these algorithms on various sizes of CM-5 and MP-1, both in terms of number of PEs in the array and local memory available within each PE. Contrary to the results reported in [18], we claim that for a single probe of the recognition phase, machine sizes larger than S would deteriorate the time performance. This is due to the fact that interconnection networks with larger diameter takes more time to perform global operations. We also show, in algorithms D, larger size machines can be used for concurrent processing of multiple probes of the recognition phase.

In the following, we tabulate our results for partitioning algorithms A, B, C, and D. Raw timing data are included in the Appendix A. Table 1 presents execution times of various subtasks using partitioning algorithms described in the previous section. The Algorithm A addresses the congestion and contention problem by computing the keys in the control processor/array control unit at the cost of redundant processing in each processor in the array. As shown in Table 1, for Algorithm A, the computation time in the control processor becomes the dominating factor in the overall execution time. We did not execute Algorithm A on MP-1 because of the large computation time and insufficient memory available within the control unit. We are unable to execute Algorithm B on various size of MP-1 as the smallest size MP-1 consists of 1K processors. The performance of Algorithms A, B, and C on various sizes of CM-5 and MP-1 is shown in Table 2.

Larger MP-1s have been used for concurrent processing of multiple probes (Algorithm D) and results are reported in Table 3. Several interesting observations on the interplay between various components of the MP-1 architecture can be made from this table. As the number of PEs increases with the number of concurrent probes, it affects various components of the execution time. For example, larger size machines mean larger diameter implying more time for global operations. On the other hand, larger machine size reduces the load on each processor, hence less time is spent on local operations.

Figures 12 and 14 show the performance of Algorithms B and C. The bin access time and voting time reduces linearly as the number of copies of the hash table in the processor array increases. The hash bin access time refers to the time taken to access hash bins corresponding to feature points in the scene. The voting time corresponds to routing the information within each voted bin, (model, basis) pairs, to compute local maximum for each pair. In Figs. 13 and 15, we simulate worst-case and semiworst-case scenario. For the worst-case, we assume that all the scene points hash to locations stored in a single PE and in the semiworst-case, all the keys hash to locations stored in a small subset of PEs. The results show the performance of various partitioning strategies adopted in algorithms B and C. In the case of hash bin access, the access time decreases linearly with the increase in the number of hash table copies resident in the processor array. On the other hand, beyond a certain number of copies of the hash table, the voting time starts increasing (see Fig. 15). This is due to the increased network traffic generated by larger number of copies.

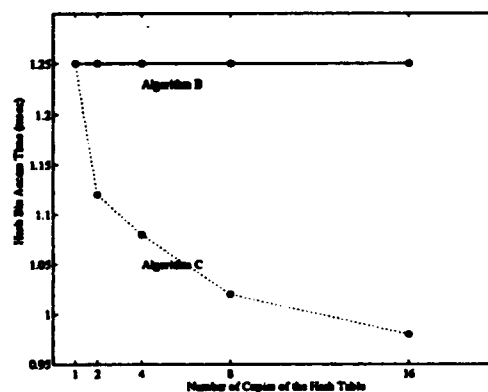


Figure 12: Hash bin access time vs Number of hash table copies for Algorithm B and Algorithm C on a 512 PE CM-5.

In Table 4, we compare our results with those reported in [1; 18]. We assume no hash table folding, symmetries, and or partial histogramming on the hash table data. Our serial implementation shows that one probe of the recognition phase takes about 13.4 seconds on a SUN SPARC2 operating at 25MHz and 32 Mbytes of on board RAM.

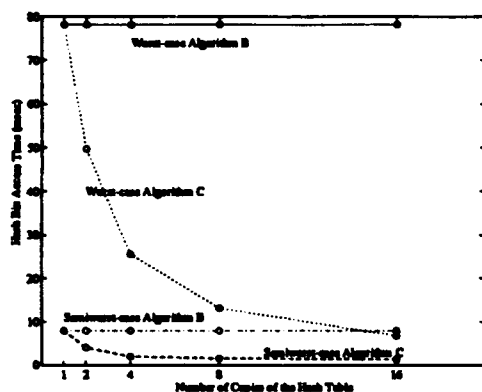


Figure 13: Hash bin access time vs Number of hash table copies for Algorithm B and Algorithm C simulating worst-case scenario on a 512 PE CM-5.

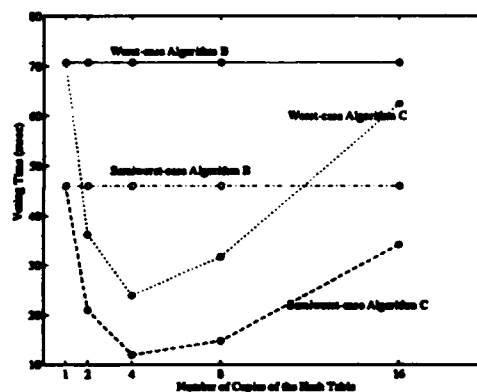


Figure 15: Voting time vs Number of hash table copies for Algorithm B and Algorithm C simulating worst-case scenario on a 512 PE CM-5.

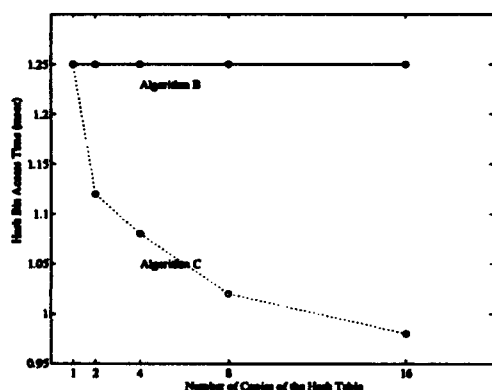


Figure 14: Voting time vs Number of hash table copies for Algorithm B and Algorithm C on a 512 PE CM-5.

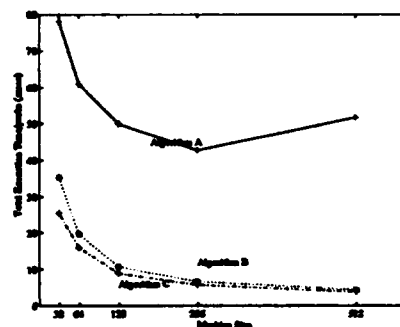


Figure 16: Total execution time vs CM-5 Machine size for various algorithms.

4.4.1 Performance Comparison: MP-1 vs CM-5

During the implementation of geometric hashing algorithm, we experimented with various aspects of the MP-1 and CM-5. Usually, SIMD machines employ fine grained massive parallelism and computationally less powerful processing elements. In MP-1, we could access machines with upto 16K processors. However, each processor has a 4-bit ALU. It takes 2.51 msec to encode a scene point. The encoding process comprises of approximately 7 floating point operations and 5 integer arithmetic operations. On the other hand, SPMD (synchronized MIMD) machines employ coarse grain parallelism with powerful processors as processing nodes. It takes 0.0825 msec to encode a scene point. However, communication intensive subtasks perform poorly on CM-5. For example, computing maximum of data elements, one element per processor, takes 0.32 msec on a 512 processor CM-5, and it takes 0.055 msec on a 1024 processor MP-1. If communication pattern is irregular, such as

in the voting process, performance of SIMD machines degrades drastically. Such computation and communication characteristics suggest the use of SIMD and SPMD machines in applications with varied characteristics. Traditionally, SIMD machines are known to be well-suited for low level vision operations. However, MP-1, with an additional router network motivates the use of MP-1 for applications with moderate computational needs and regular global communication patterns. Several heuristic techniques in high level vision fall in this category. SPMD machines, such as CM-5, are suitable for applications with high computational needs and moderate global communication requirements. In addition, in the absence of efficient data partitioning and routing techniques, the performance of such machines degrades for applications with local neighborhood communication requirements. The memory available with each processor also affects the usage of the underlying architecture. Traditionally, due to limitations of VLSI and cost considerations, memory available within each processor is relatively less in SIMD machines, compared with SPMD machines. Limited memory can affect the performance

of applications which require storage of large volume of on-line data.

5 Conclusion

We have presented scalable data parallel algorithms for geometric hashing. Based on these algorithms, we have obtained fast parallel implementations. The implementations achieve much superior time performance than those known in the literature. These implementations are developed after carefully studying the characteristics of the underlying architectures of CM-5 and MP-1, i.e. *fat tree* and mesh array, respectively. Various experiments were conducted to fine tune the partitioning and the mapping strategies to suit the communication and the computation capabilities of these machines. Based on these experiments, data parallel algorithms were designed to efficiently utilize the architectural and programming features. This experimentation has assisted in achieving uniform distribution of work load in the machines during the execution of algorithms leading to fast and scalable implementations. Our early work in using CM-5 and MP-1 for high level vision is very encouraging and brings a promising future to the applications of parallel processing techniques in realizing real time integrated vision systems.

References

- [1] O. Bourdon and G. Medioni. Object recognition using geometric hashing on the Connection Machine. In *International Conference on Pattern Recognition*, pages 596-600, 1988.
- [2] Charles E. Leiserson et.al. The network architecture of the Connection Machine CM-5. Technical report, Thinking Machines Corporation, 1992.
- [3] W. E. L. Grimson. On the recognition of parameterized 2d objects. *International Journal of Computer Vision*, pages 2(4):353-372, 1989.
- [4] W. E. L. Grimson and D. P. Huttenlocher. On the verification of hypothesized matches in model-based recognition. In *First Europe Conference on Computer Vision*, pages 489-498, 1990.
- [5] W. E. L. Grimson and T. L. Perez. Localizing overlapping parts by searching the interpretation trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):469-482, 1987.
- [6] D. P. Huttenlocher and S. Ullman. Object recognition using alignment. In *IEEE First International Conference on Computer Vision*, pages 102-111, 1987.
- [7] A. Khokhar. *Scalable Data Parallel Algorithms and Implementations for Object Recognition*. PhD thesis, Department of EE-Systems, University of Southern California, Los Angeles, January 1993.
- [8] A. Khokhar, H-J. Kim, and V. Prasanna. Scalable geometric hashing on MasPar MP-1. submitted to *International Conference on Computer Vision and Pattern Recognition*, New York, NY, 1993.
- [9] A. Khokhar and W. Lin. Stereo and image matching on fixed size linear arrays. In *International Parallel Processing Symposium*, Newport Beach, CA, 1993.
- [10] A. Khokhar, W. Lin, and V. K. Prasanna. Stereo and image matching on fixed size mesh arrays. In *International Conference on Computer Architectures and Machine Perception*, Paris, France, December 1991.
- [11] A. Khokhar, V. Prasanna, and C. Wang. Object recognition using geometric hashing on the Connection Machine CM-5. Technical report, Department of EE Systems, University of Southern California, Los Angeles, CA, September 1992.
- [12] V. K. Prasanna Kumar. *Parallel Algorithms and Architectures for Image Understanding*. Academic Press, 1991. Edited Volume.
- [13] Y. Lamdan and H. J. Wolfson. Geometric hashing: A general and efficient model based recognition scheme. In *International Conference on Computer Vision*, pages 218-249, Tampa, FL, December 1988.
- [14] Charles E. Leiserson. FAT-TREES:universal networks for hardware efficient supercomputing. In *International Conference on Parallel Processing*, pages 393-402, 1985.
- [15] W. Lin and V. K. Prasanna Kumar. Efficient histogramming on hypercub SIMD machines. *Computer Vision, Graphics, and Image Processing*, 49(1):104-120, January 1990.
- [16] P. N. Pani, A. Khokhar, and V. K. Prasanna. Parallel algorithms for stereo based on discrete relaxation techniques. In *International Conference on Pattern Recognition*, Amsterdam, Holland, August 1992.
- [17] Lutz Prechelt. Measurements of MasPar MP-1261a communication operations. Technical Report DXX/93, Institut für Programmstrukturen und Datenorganisation, Universität Karlsruhe, Postfach 6980, D-7500 Karlsruhe, Germany, 1993.
- [18] I. Rogoutsos and R. Hummel. Implementation of geometric hashing on the Connection Machine. In *IEEE CAD-Based Vision Workshop*, pages 76-84, Maui, HI, 1991.
- [19] F. Stein and G. Medioni. Efficient two dimensional object recognition. In *International Conference on Pattern Recognition*, pages 13-17, Ann Arbor, MI, June 1990.
- [20] H. J. Wolfson. Model based object recognition by geometric hashing. In *First Europe Conference on Computer Vision*, pages 526-536, 1990.

Partitioning Algorithm	Machine Type	Encoding Scene Points	Hash Bin Access	Voting	Computing Local Maximum of Votes	Computing Global Maximum of Votes	Total Time
A	CM-5	33.3	5.45	1.04	1.83	0.29	41.45
B	CM-5	0.33	1.96	2.27	1.83	0.29	6.68
C	CM-5	0.33	1.33	1.96	1.83	0.29	5.74
B	MP-1	2.51	18.53	24.10	3.36	0.055	48.76
C	MP-1	2.51	6.78	20.02	3.36	0.055	32.72

Table 1: Execution times (in msec) of various subtasks in a probe using different partitioning algorithms for a scene consisting of 1024 feature points on a 256 CM-5 and 1K MP-1.

Machine Size/Type	Algorithm A (in msec)	Algorithm B (in msec)	Algorithm C (in msec)
32/CM-5	78.07	35.38	25.48
64/CM-5	61.04	19.75	16.02
128/CM-5	50.02	10.77	8.96
256/CM-5	41.45	6.68	5.74
512/CM-5	51.74	4.41	3.8
1K/MP-1	XX	48.76	32.72

Table 2: Execution times (in msec) of various algorithms on a scene consisting of 1024 feature points.

Number of Probes	Machine Size	Encoding Scene Points	Hash Bin Access	Voting	Computing Local Max of Votes	Computing Global Max of Votes	Total Time
1	1K	2.51	18.53	24.10	3.36	0.055	48.76
2	2K	2.51	23.16	30.60	8.16	0.165	64.59
4	4K	2.51	40.36	49.86	8.16	0.314	101.20
8	8K	2.51	54.62	49.72	8.16	0.608	115.62

Table 3: Execution times (in msec) of Algorithm C on a scene consisting of 1024 feature points with concurrent processing of multiple probes on various sizes of MP-1. Average bin size is 8.

Methods	# of Models (16 points/model)	Machine ^a Size/Type	# of Scene Points	Total Time
Our Method (A)	1024	256/CM5	1024	42.76 msec
Our Method (B)	1024	256/CM-5	1024	6.68 msec
Our Method (C)	1024	256/CM-5	1024	5.74 msec
Our Method (B)	1024	1K/MP-1	1024	53.37 msec
Our Method (C)	1024	1K/MP-1	1024	38.89 msec
Our Method (B)	1024	1K/MP-1	200	49.50 msec
Our Method (B)	1024	256/CM-5	200	4.50 msec
Hummel et. al.[18]	1024	8K/CM-2	200	800 msec
Medioni et. al. [1]	x	8K/CM-2	x	2.0-3.0 sec

Table 4: Comparison with previous implementations.

^aEach processor in CM-5, CM-2, and MP-1 operates at 32MHz, 7MHz, and 12.5MHz respectively.

An Integrated Object Recognition System Based on High Degree Implicit Polynomials, Algebraic Invariants, and Bayesian Methods *

Jayashree Subrahmonia, Daniel Keren and David B. Cooper

Laboratory for Engineering Man/Machine Systems,

Division of Engineering, Brown University, Providence, RI 02912, USA

email: js@lems.brown.edu

Abstract

This paper presents a new robust low-computational-cost system for recognizing freeform objects in 3D range data or in 2D curve data in the image plane. Objects are represented by implicit polynomials (i.e., 3D algebraic surfaces or 2D algebraic curves) of degrees greater than 2, and are recognized by computing and matching vectors of their algebraic invariants (which are functions of their coefficients that are invariant to translations, rotations, and general linear transformations). Implicit polynomials of 4th degree can represent complicated asymmetric free-form shapes. This paper deals with the design of Bayesian (i.e., minimum probability of error) recognizers for these models and their invariants that results in low computational cost recognizers that are robust to noise, *partial occlusion*, and other perturbations of the data sets. This work extends the work in [4] by developing and using new invariants for 3D surface polynomials and applying the Bayesian recognizer to operating on invariants. The recognizer seems to be ideally suited to robot vision, handprinted character recognition, ATR when used with Kimia's partitioning algorithms [10, 5], and other applications.

1 Introduction

The simplest 2D or 3D recognition problem is that a boundary model is stored in a database for each of L rigid objects. (By an object *boundary*, we mean the 3D object surface, and external and internal boundary curves for a 2D object.) Data along the entire boundary or over a portion of the boundary of an object to be recognized is collected from a sensor. Object recognition is to be realized by determining the stored boundary model that fits the sensed data the best. By best, we mean in the sense of minimum mean squared distance from the data points to a boundary model. This should produce the recognizer functioning with the highest relative frequency of correct recognition. What are the drawbacks to this approach? There are two, both computational. First is that, if there are N data points, order of NL computations must be made for checking on the mean square fits of L stored object boundaries to N data points. This can be considerable if L is large. Second is that the position of the object being sensed will be different than the position of the object in the

database and the sensed data may also have undergone a general linear transformation, due, e.g., to the viewing of a target boundary on the ground from an arbitrary aerial viewing direction. Hence, an object in the database has to be rotated and translated in checking its match to the sensed data. This usually means checking a boundary model fit to the data for the boundary model moved to many different positions and linearly transformed, thus incurring a huge amount of computation. The approach presented in this paper avoids both these drawbacks.

2 Recognition Approach

The approach in this paper is to model 3D and 2D objects of interest by algebraic surfaces or curves, respectively, i.e., by the *zero sets* of implicit polynomials. The *zero set* is the set of points (x, y) in 2D (or (x, y, z) in 3D) for which the polynomial function $f(x, y)$ on 2D (or $f(x, y, z)$ on 3D) is zero. Then, a stored model is simply the set of coefficients for the polynomial model. These are global 3D models, unlike explicit polynomials where z is given as an explicit function of x and y as in a depth map. Most of the early work on implicit polynomial curves and surfaces was limited to quadrics, thus dealing with representations that had modest expressive power. Implicit polynomials of degree greater than 2, on the other hand, have great modeling power for complicated objects and can be fit to data very well. In [7], Taubin, formerly of our laboratory, has presented a very well organized and understandable introduction to these polynomials and some of their properties, and developed very effective approaches to low computational cost algorithms for fitting these polynomials. Hence we can now use these high degree implicit polynomials for representing complicated objects. In addition, we have developed a technique for fitting polynomials with bounded zero sets, which results in better and more stable description of objects [3].

For the implicit polynomial models, checking the fit of a stored surface or curve to data involves fitting an implicit polynomial to the data, and then comparing the resulting polynomial coefficients with the L coefficient vectors (one for each object) stored in the database. If the object to be recognized is in a different position than the object in the database, the coefficients for the best fitting polynomial to the data will be different than the coefficients for the

*This work was partially supported by NSF Grant #IRI-8715774 and NSF-DARPA Grant #IRI-8905436

same object in the database. Our solution to this problem is to use a vector of algebraic invariants for a recognizer. An algebraic invariant is a function of the implicit polynomial coefficients that is invariant to rotations and translations for 3D surfaces and is invariant to translations and general linear transformations for 2D curves. The required computation for comparing the set of coefficients or the set of invariants is roughly the order of the number of data points (which is the amount of computation required to fit an implicit polynomial to the data set). Unfortunately, the situation is not quite so simple, and a problem arises here. This paper presents the solution to the problem and the resulting recognizer.

The problem that had to be solved is that small changes in a data set often result in large changes in the coefficients of the best fitted polynomial, and, hence, large changes in the algebraic invariants. The reason for this variability is due to the fact that the data used in fitting the polynomials provides constraints among the coefficients of a fitted polynomial, but the data may be insufficient to uniquely determine the coefficients. Hence, since the fitted curve and stored curve coefficients may differ greatly, we cannot compare the curves over the local region of interest based on their coefficients or the invariants, which are functions of these coefficients. Our solution to this problem is to treat recognition as Bayesian statistical recognition in the presence of noisy data, and to use certain asymptotic results which permit a computationally low cost recognizer. The resulting recognizer involves comparison of the measured vector of invariants, but not using the Euclidean distance. *Rather, the error measure requires the use of a weighting matrix which is a function of the specific data set being recognized.* The required computation for computing this matrix is of the order of the number of data points, and hence, modest and suitable for real time recognition. The beauty of this approach is that even though it uses global models — implicit polynomials and coefficient vectors — it behaves as though recognition is based on the matching of a local data set to a boundary model. *Hence, it works excellently even if the data set is over only a portion of the object boundary, which will be the case due to self occlusion if range data is taken for a 3D object from one direction, or which may be the case if one or more objects are partially occluding the object to be recognized.*

The known invariants in the mathematical literature are affine invariants (i.e., quantities that are invariant under translations, rotations and stretchings along the x , y and z axes) that are functions of only the leading form. The leading form is the part of the polynomial that contains terms of the highest degree. For example, $a_{20}x^2 + a_{11}xy + a_{02}y^2$ is the leading form of the second degree implicit polynomial $f(x, y) = a_{20}x^2 + a_{11}xy + a_{02}y^2 + a_{10}x + a_{01}y + a_{00}$, in 2D. We extend this in [8, 2, 4, 9] where new large classes of affine and Euclidean invariants of all the coefficients in a polynomial are introduced. An example of a new affine invariant for a fourth degree polynomial in x, y, z , $f(x, y, z) = \sum_{i+j+k \leq 4} a_{ijk}x^i y^j z^k$, found using our symbolic method for discovering new algebraic in-

variants [4, 2], is

$$\begin{aligned} & 216a_{121}^2 a_{202} - 648a_{112}a_{130}a_{202} + 1296a_{040}a_{202}^2 - \\ & 108a_{112}a_{121}a_{211} + 972a_{103}a_{130}a_{211} - 648a_{031}a_{202}a_{211} + \\ & 216a_{022}a_{211}^2 + 216a_{112}^2 a_{220} - 648a_{103}a_{121}a_{220} + \\ & 432a_{022}a_{202}a_{220} - 648a_{013}a_{211}a_{220} + 1296a_{004}a_{220}^2 - \\ & 3888a_{040}a_{103}a_{331} + 972a_{031}a_{112}a_{301} - 648a_{022}a_{121}a_{301} + \\ & 972a_{013}a_{130}a_{301} + 972a_{031}a_{103}a_{310} - 648a_{022}a_{112}a_{310} + \\ & 972a_{013}a_{121}a_{310} - 3888a_{004}a_{130}a_{310} + 1296a_{022}^2 a_{400} - \\ & 3888a_{013}a_{031}a_{400} + 15552a_{004}a_{040}a_{400} \end{aligned}$$

Experimental results on using these invariants for object recognition are given in section 4.

3 Asymptotic parameter distribution and Bayesian Recognition

Let α denote the vector of coefficients of the polynomial $f(x, y, z)$ that describes the given object. We assume that the range data points Z_1, Z_2, \dots, Z_N are statistically independent, with Z_i having probability density function (pdf)

$$p(Z_i | \alpha) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{1}{2\sigma^2} \frac{f^2(Z_i)}{\|\nabla f(Z_i)\|^2} \right] \quad (1)$$

The assumption is that Z_i is a noisy Gaussian measurement of the object surface in the direction perpendicular to the boundary at its closest point. This model is introduced and discussed in [1, 6, 4]. Thus, the joint probability of the data points is

$$p(Z^N | \alpha) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp \left[-\frac{1}{2\sigma^2} \sum_{i=1}^N \frac{f^2(Z_i)}{\|\nabla f(Z_i)\|^2} \right] \quad (2)$$

Being able to write this joint probability for a data set in terms of a complicated curve or surface is an important result and permits the application of a large range of tools from statistics and probability theory. The maximum likelihood estimate $\hat{\alpha}_N$ of α given the data points is the value of α that maximizes (2). A very useful tool for solving the problems of object recognition and parameter estimation is an asymptotic approximation to the joint likelihood function, (2), which can be shown to have a Gaussian shape in α [1, 6], i.e.,

$$p(Z^N | \alpha) \approx [p(Z^N | \hat{\alpha}_N)] \exp \left\{ -\frac{1}{2} (\alpha - \hat{\alpha}_N)^t \Psi_N (\alpha - \hat{\alpha}_N) \right\} \quad (3)$$

where Ψ_N is the second derivative matrix having i, j th component $-\frac{\partial^2}{\partial \alpha_i \partial \alpha_j} \ln p(Z^N | \alpha) |_{\alpha=\hat{\alpha}_N}$. Hence, all the useful information about α is summarized in the quadratic form in the exponent of equation (3). If Ψ_N is not singular, then it is the inverse covariance matrix of $\hat{\alpha}_N$. The matrix Ψ_N is called the Fisher Information matrix of $\hat{\alpha}_N$. Various extremely useful generalizations of (3) are developed in [6].

The asymptotic approximation (3) gives an understanding of the extent to which the data constrains the coefficients of the best fitting polynomial [4].

The next section deals with using this approximation for designing a metric based on the geometric invariants for comparing two polynomial zero sets over the region where the data exists.

3.1 Mahalanobis distance between two sets of Invariants

The scenario for recognition that we consider in this paper is one where we have a set of objects labeled $l = 1, 2, \dots, L$ in the database, all modeled by polynomials of the same degree. Let G_l denote the vector of invariants for the polynomial describing object l . Then, given a set of range data points, $Z^N = \{Z_1, Z_2, \dots, Z_N\}$, the optimum recognition rule is 'choose l for which $p(Z^N | G_l)$ is maximum'. Thus, the recognition problem reduces to computing the likelihood of the data given G . In [6], we have shown that

$$p(Z^N | G) \approx [p(Z^N | \hat{\alpha}_N)] (2\pi)^{-\frac{d_H}{2}} |\Psi_N^H|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(G - \hat{G}_N)^t \Psi_N^G (G - \hat{G}_N)\} \quad (4)$$

where Ψ_N^G and Ψ_N^H are the Information matrices of the vector of invariants and a vector of nuisance parameters, respectively, and d_H is the number of nuisance parameters.

Using (4) for the simplest case of recognition, the optimum recognition rule becomes - 'Choose l for which the Mahalanobis distance, $(G_l - \hat{G}_N)^t \Psi_N^G (G_l - \hat{G}_N)$, is minimum.' This is because, the only part of (4) that is a function of l is $\exp\{-\frac{1}{2}(G_l - \hat{G}_N)^t \Psi_N^G (G_l - \hat{G}_N)\}$.

The beauty of this recognizer is that the computational cost is negligible, but the recognizer is equivalent to checking how well the data fits the models stored in the database for different linear transformations of the models for which the computational cost is enormous.

In summary, object recognition using invariants is done as follows.

1. Fit the best polynomial to the data set.
2. Compute the invariants \hat{G}_N which are functions of the coefficients of the polynomial.
3. Compute the Mahalanobis distance, $(G_l - \hat{G}_N)^t \Psi_N^G (G_l - \hat{G}_N)$ to each object in the database and pick the l for which it is a minimum.

This computational cost for step 1 is linear in the number of data points, and typically is a fraction of a second on a SPARC 10 for 200 data points and a 4th degree implicit polynomial curve. In step 2, invariants such as that in section 2 must be computed. Computation time for 5 to 10 of these is less than that for step 1. The only time consuming computation in step 3 is the computation of Ψ_N^G . This matrix is given by

$$\Psi_N^G = (DG)^\dagger |_{\alpha=\hat{\alpha}_N} \Psi_N (DG)^\dagger |_{\alpha=\hat{\alpha}_N}$$

where \dagger implies pseudo-inverse and $D(G)$ is the Jacobian of the transformation from α to G . The

costly computation here is that for Ψ_N which is linear in the number of data points and is of the order of the computation in step 1. Hence, the computation of the Mahalanobis distance for 200 data points and a 4th degree curve is a fraction of a second on a SPARC 10 and can be sped up by orders of magnitude with parallel architectures.

Experimental results illustrating the use of the Mahalanobis distance for recognition are given in the next section.

4 Experimental Results

The experiments illustrate the use of the Mahalanobis distance in the space of invariants for recognizing 2D and 3D objects from real data that may be partial and that is noisy. The experiments also illustrate the fact that the Mahalanobis distance has better discriminatory power than does the Euclidean distance.

The first set of experiments illustrate the performance of the recognizer for 3D objects. The objects in this experiment are keyboard mice. Figure 1 shows the four mice used in this experiment. Figures 2(a)-(d) are the data sets and the polynomial fits for the mice in standard position. (The polynomial fits were obtained using our approach for fitting bounded polynomials). The data sets were obtained using the Brown and Sharpe Microval Manual coordinate measuring machine. All the data sets are well fit by fourth degree polynomials in x, y, z . These are the four objects in the database.

Figures 3(a)-(d) are the data sets and polynomial fits for the rotated and translated versions of the mice in the database. We used 7 invariants for a fourth degree polynomial in x, y, z . All of them are listed in [2]. The goal in this experiment is to recognize the mice in Figures 3(a)-(d) using the Mahalanobis distance measure and compare the results with those using the Euclidean distance.

Tables 1 and 2 show the Mahalanobis and the Euclidean distances, respectively, between the vector of invariants for the polynomial fits to the rotated mice in Figure 3 and the vectors of invariants for the four mice in the database. The Mahalanobis distance measure does a great job of discriminating between the right object and the rest. Also, the Mahalanobis distance has much better discriminatory power than does the Euclidean distance.

The next experiment illustrates the use of the Mahalanobis distance for recognizing 2D and 3D objects from partial data.

Figure 4 shows the partial data (with the polynomial fit superimposed) for the mouse in Figure 2(a). The partial data in this experiment is what a stereo sensor would see when looking at the mouse from a point near the bottom left corner. The Mahalanobis and Euclidean distances between the vector of invariants for the polynomial fit to the occluded object and the stored vectors of invariants are :

Mahalanobis distance:
 Mouse1 : 1.0 Mouse2 : 1065
 Mouse3 : 30.31 Mouse4 : 1.004
 Euclidean distance:
 Mouse1 : 1.0 Mouse2 : 18.39
 Mouse3 : 1.619 Mouse4 : 0.901

The Mahalanobis distance to Mouse1 is the smallest. However, the Mahalanobis distance to Mouse4 is almost the same as that to Mouse1. This is because the occluded data does not contain the curved front part of Mouse1, and since that is the part that really distinguishes Mouse1 from Mouse4, it is hard to distinguish between them based on the partial data. The distances to Mouse2 and Mouse3 are large compared to those to Mouse1 and Mouse4. The Euclidean distance does not give good recognition results with partial data. In fact, the Euclidean distance from the occluded object to Mouse4 is smaller than that to Mouse1.

The data sets for the 2D examples are handwritten characters. The objects in the database are the handwritten characters, 'a', 'q', 'g' and 'w', shown in Figures 5(a)-(d). The data sets are well fit by fourth degree polynomials in x, y . Figure 6(a) is another instance of the handwritten character 'w' that is a rotated, translated, occluded and noisy version of the one in the database. We fit a fourth degree polynomial to the occluded object in order to compare its invariants with those for the unoccluded database objects. Figure 6(b) is the fourth degree polynomial fit to the data set in 6(a). Three invariants for a fourth degree polynomial in x, y obtained using our approach are

1. $3a_{13}^2 - 8a_{04}a_{22} + 2a_{13}a_{31} + 3a_{31}^2 - 32a_{40}a_{04} - 8a_{22}a_{40},$
2. $3a_{04}^2 + 2a_{04}a_{22} + a_{13}a_{31} + 2a_{04}a_{40} + 2a_{22}a_{40} + 3a_{40}^2,$
3. $a_{22}^2 - 3a_{13}a_{31} + 12a_{04}a_{40},$

Since the invariants should be independent of multiplication of the coefficients by a constant, these three functions yield only two invariants. One set of two invariants is $\frac{a_{13}}{a_{31}}$ and $\frac{a_{22}}{a_{31}}$. Thus, for object recognition, we use the Mahalanobis distance between the ratios of invariants. The Mahalanobis distance from the letter in Figure 6(a) to the letters 'a', 'g', 'q' and 'w' in the database are:

'a':1.00 'q':12.9 'g':12.2 'w':4.81

(The distance to 'a' in the database is normalized to have value 1.0.) The distance to 'w' is minimum. However, the distance is small to 'a' and 'q'. This is because, the data set in 6(a) also fits the model for 'a' and 'q' as shown in Figures 6(c) and 6(d). The experiment illustrates that even under a large amount of occlusion, the recognizer comes up with the best possible results.

References

- [1] R.M. Bolle and D.B. Cooper. On Optimally Combining Pieces of Information, With Applications to Estimating 3D Complex-object Position From Range Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):619-638, September 1986.
- [2] D. Keren. Some New Invariants in Computer Vision. 1992. Under review for publication in *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- [3] D. Keren, D. B. Cooper, and J. Subrahmonia. Describing Complicated Objects by Implicit Polynomials. Technical Report LEMS-102, Brown University, February 1992. Accepted for publication as a regular paper in the *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- [4] D. Keren, J. Subrahmonia, D. B. Cooper, and G. Taubin. Bounded and Unbounded Implicit Polynomial Curves and Surfaces, Mahalanobis Distances, and Geometric Invariants, for Robust Object Recognition. In *Proceedings: Image Understanding Workshop*, pages 769-778, San Diego, January 1992.
- [5] Kaleem Siddiqi and Benjamin B. Kimia. Parts of visual form: Computational aspects. In *CVPR*, 1993.
- [6] J. Subrahmonia, D. B. Cooper, and D. Keren. Practical Reliable Bayesian Recognition of 2D and 3D Objects Using Implicit Polynomials and Algebraic Invariants. Technical Report LEMS-107, Brown University, May 1992. Under review for publication in the *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- [7] G. Taubin. Estimation of Planar Curves, Surfaces and Nonplanar Space Curves Defined by Implicit Equations, with Applications to Edge and Range Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:1115-1138, November 1991.
- [8] G. Taubin and D. B. Cooper. 3D Object Recognition and Positioning with Algebraic Invariants and Covariants. pages 147-182, July 1990. A chapter in *Symbolic and Numerical Computations-Towards Integration*, pages 147-182, B. R. Donald, D. Kapur and J. Mundy editors, Academic Press, 1992.
- [9] G. Taubin and D. B. Cooper. 2D and 3D Object Recognition and Positioning System Based on Moment Invariants. In *Proceedings of the DARPA-ESPIRIT Workshop on Geometric Invariants*, Rikjavik, Iceland, pages 235-258, May 1991. Also in book *Geometric Invariance in Machine Vision*, pages 375-397, J. Mundy and A. Zisserman editors, MIT Press, 1992.
- [10] Kathryn J. Tresness, Kaleem Siddiqi, and Benjamin B. Kimia. Parts of visual form: Ecological and psychophysical aspects. Technical Report LEMS 104, LEMS, Brown University, May 1992.

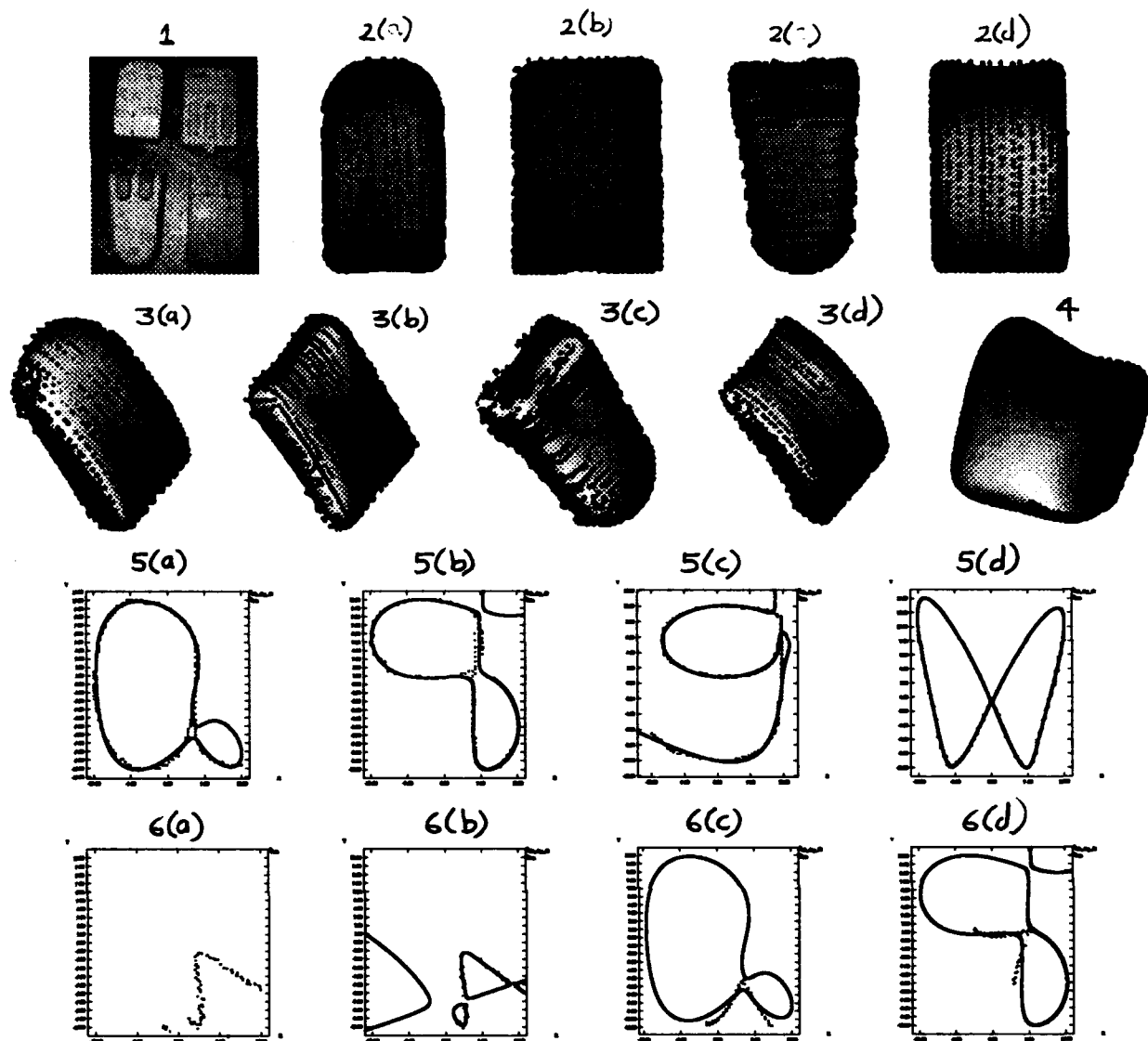


TABLE 1 (Mahalanobis distances) :

	Mouse1 (rotated)	Mouse2 (rotated)	Mouse3 (rotated)	Mouse4 (rotated)
Mouse1	1.000e+00	3.517e+02	1.379e+01	3.519e+00
Mouse2	2.463e+01	1.000e+00	1.560e+02	2.109e+01
Mouse3	2.622e+02	3.717e+03	1.000e+00	1.489e+02
Mouse4	2.872e+00	2.818e+03	6.096e+01	1.000e+00

TABLE 2 (Euclidean distances) :

	Mouse1 (rotated)	Mouse2 (rotated)	Mouse3 (rotated)	Mouse4 (rotated)
Mouse1	1.000e+00	2.535e+00	9.614e+00	2.741e+00
Mouse2	1.822e+00	1.000e+00	1.125e+00	1.547e+00
Mouse3	1.351e+01	2.322e+00	1.000e+00	7.568e+00
Mouse4	1.446e+00	1.327e+00	2.435e+01	1.000e+00

Reflectance Based Recognition

Shree K. Nayar and Ruud M. Bolle

Department of Computer Science, Columbia University, New York, N.Y. 10027 *

Abstract

Neighboring points on a smoothly curved surface have similar surface orientations and illumination conditions. Hence, their brightness values can be used to compute the ratio of their reflectance coefficients. Based on this observation, we develop an efficient algorithm that estimates a reflectance ratio for each region in an image with respect to its background. The region reflectance ratio represents a physical property of a region that is invariant to the illumination conditions. The ratio invariant is used to recognize objects from a single brightness image of a scene. We conclude with experimental results that demonstrate the power of using reflectance and geometric properties of objects, simultaneously.

1 Introduction

Object recognition has been an active area of machine vision research for the past two decades [1]. The traditional approach has been to recover geometric features from images and then use these features to hypothesize and verify the existence of three-dimensional objects in the image. Edges and vertices are examples of geometric features often used by recognition systems. In the past, little attention has been given to the use of other physical properties of objects for recognition. In addition to its geometry, an object may be characterized by intrinsic properties such as reflectance, roughness, and material type. Clearly, the representation of an object using all of these intrinsic properties is useful only if the recognition system is able to compute the properties from images.

In this paper, we present a method for computing the reflectance of regions in a scene, with respect to their backgrounds, from a single image. The result is a physical property of each scene region that is invariant to the intensity and direction of illumination. This photometric invariant, referred to as the *reflectance ratio*, provides valuable information for recognition tasks. The reflectance ratios (photometric features) of object regions and the spatial configuration (geometric features) of the regions are used to represent the object.

The problem of computing the reflectance of regions in a scene was first addressed by Land [2]. In general, image brightness is the product of surface reflectance and illumination. Land developed the retinex theory that suggests computational steps for recovering the reflectance (or lightness) of scene regions in the presence of varying illumination. Subsequently, several hardware implementations for the retinex theory were proposed [3], [4]. The main idea underlying Land's lightness computation is global consistency. The lightness value computed for any particular region must be consistent with those computed elsewhere in the image. However, realistic images include shadows, occlusions, and noise. Each one of these factors can cause a region boundary to go undetected or the computed lightness of a region to be erroneous. Such errors can greatly affect the lightness values computed for all other regions in the image. For this reason, Land's global method is not applicable to most real images.

In this paper, we develop an alternative scheme for computing the ratio of the reflectance of a region to that of its background. The image is first segmented into regions of constant (but unknown) reflectance. Next, a reflectance ratio is computed for each region and its background using only points that lie close to the region's boundary. In this approach, the reflectance ratio computed for any particular region is not affected by those computed for regions elsewhere in the image. Land's analysis [2] was restricted to planar (two-dimensional) scenes with patches of constant reflectance. In contrast, our derivation of the reflectance ratio is based on the analysis of regions that lie on curved surfaces. In the case of curved surfaces, image brightness variations result from both illumination variations as well as surface normal changes. For curved surfaces, our reflectance ratio invariant is valid when a region and its background have the same distribution (scattering) function but different reflectance coefficients (albedo).

Recently, Finlayson [5] proposed computing histograms using ratios in different color channels for object recognition. Histograms, however, are in general sensitive to the scale and rotation of objects in the scene and hence are not effective for three-dimensional object recognition and pose estimation. Here, we use the re-

*This research was supported in part by DARPA Contract No. DACA 76-92-C-0007 and in part by the David and Lucile Packard Fellowship. R. M. Bolle is supported by the IBM T.J. Watson Research Center, Yorktown Heights, N.Y. 10598, U.S.A.

reflectance ratio invariant to recognize objects from a single image. This approach is very effective in the case of man-made objects that have printed characters and pictures. Each object is assumed to have a set of regions, each with constant reflectance. The reflectance ratio and center of each region are used to represent objects using a hash table. Recognition and pose estimation algorithms are presented that use the reflectance ratios of scene regions to index the hash table. The result is a hypothesis for the existence of an object in the image. This hypothesis is verified using the reflectance ratios and locations of other regions in the scene. Recognition results are presented for realistic scenes with occlusion, shadows, and illumination variations. These results show the simultaneous use of reflectance and geometry to be a powerful approach to object recognition problems.

2 Reflectance Ratio Invariant

The reflectance of a surface depends on its roughness and material properties. In general, incident light is scattered by a surface in different directions. This distribution of reflected light can be described as a function of the angle of incidence, the angle of emittance, and the wavelength of the incident light. Consider an infinitesimal surface patch with normal \mathbf{n} , illuminated with monochromatic light of wavelength λ from the direction \mathbf{s} , and viewed from the direction \mathbf{v} . The reflectance of the surface patch can be expressed as: $r(\mathbf{s}, \mathbf{v}, \mathbf{n}, \lambda)$. Now consider an image of the surface patch. If the spectral distribution of the incident light is $e(\lambda)$ and the spectral response of the sensor is $s(\lambda)$, the image brightness value produced by the sensor is:

$$I = \int s(\lambda) e(\lambda) r(\mathbf{s}, \mathbf{v}, \mathbf{n}, \lambda) d\lambda \quad (1)$$

If we assume the surface patch is illuminated by "white" light and the spectral response of the sensor is constant within the visible-light spectrum, then $s(\lambda) = s$ and $e(\lambda) = e$. We have:

$$I = s e \rho R(\mathbf{s}, \mathbf{v}, \mathbf{n}) \quad (2)$$

where $\rho R(\mathbf{s}, \mathbf{v}, \mathbf{n})$ is the integral of $r(\mathbf{s}, \mathbf{v}, \mathbf{n}, \lambda)$ over the visible-light spectrum. We have decomposed the result into $R(\cdot)$ which represents the dependence of surface reflectance on the geometry of illumination and sensing, and ρ which may be interpreted as the fraction of the incident light that is reflected in all directions by the surface. Incident light that is not reflected by the surface is absorbed or transmitted through the surface. Two surfaces with the same distribution function $R(\cdot)$ can have different reflectance coefficients ρ .

As a result of the white-light assumption, the reflectance coefficient ρ is independent of wavelength. This enables us to represent the reflectance of the surface element with a single constant. The same can be achieved

by using an alternative approach which does not require making assumptions about the spectral distribution of the incident light and the spectral response of the sensor. Consider a narrow-band filter with spectral response $f(\lambda)$, placed in front of the sensor. Image brightness is then:

$$I = \int f(\lambda) s(\lambda) e(\lambda) r(\mathbf{s}, \mathbf{v}, \mathbf{n}, \lambda) d\lambda \quad (3)$$

Since the filter is narrow-band, it essentially passes a single wavelength λ' of reflected light. Its spectral response can therefore be expressed as:

$$f(\lambda) = \delta(\lambda' - \lambda) \quad (4)$$

The image brightness measured with such a filter is:

$$I = s' e' r(\mathbf{s}, \mathbf{v}, \mathbf{n}, \lambda') \quad (5)$$

where $s' = s(\lambda')$ and $e' = e(\lambda')$. Once again, the reflectance function can be decomposed into a geometrical function and a reflectance coefficient:

$$I = s' e' \rho' R'(\mathbf{s}, \mathbf{v}, \mathbf{n}) \quad (6)$$

In this case, $R'(\cdot)$ represents the distribution of reflected light for a particular wavelength of incident light. On the other hand, for white-light illumination, $R(\cdot)$ represents the distribution computed as an average over the entire visible-light spectrum. However, the individual terms in both (2) and (6) represent similar effects. Since we have used the white-light illumination assumption in our experiments, we will use the following expression for image brightness in our discussion:

$$I = k \rho R(\mathbf{s}, \mathbf{v}, \mathbf{n}) \quad (7)$$

The constant $k = s.e$ accounts for the brightness of the light source and the response of the sensor. The exact functional form of $R(\mathbf{s}, \mathbf{v}, \mathbf{n})$ is determined to a great extent by microscopic structure of the surface; generally $R(\cdot)$ includes a diffuse component and a specular component [6]. Once again, the reflection coefficient ρ is the fraction of incident light that is reflected by the surface. It represents the reflective power of the surface and is sometimes referred to as surface albedo.

Consider two *neighboring* points on a surface (Figure 1). For a smooth continuous surface, the two points may be assumed to have the same surface normal vectors. Further, the two points have the same source and sensor directions. Hence, the brightness values, I_1 and I_2 , of the two points may be written as:

$$I_1 = k \rho_1 R_1(\mathbf{s}, \mathbf{v}, \mathbf{n}) \quad (8)$$

$$I_2 = k \rho_2 R_2(\mathbf{s}, \mathbf{v}, \mathbf{n}) \quad (9)$$

The main assumption made in computing the reflectance ratio is that the two points have the same reflectance functions ($R_1 = R_2 = R$) but their reflectance coefficient

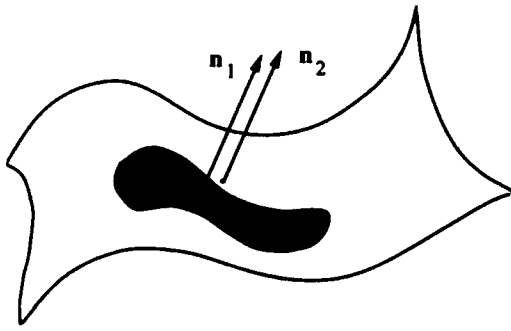


Figure 1: Neighboring points on a surface.

ρ_1 and ρ_2 may differ. An example is that of two neighboring Lambertian points that have different albedo values because they lie in regions that have different shades or colors. Then, the image brightness values produced by the two points are:

$$\begin{aligned} I_1 &= k \rho_1 R(s, v, n) \\ I_2 &= k \rho_2 R(s, v, n) \end{aligned} \quad (10)$$

The ratio of the reflectance coefficients of the two points is:

$$p = I_1/I_2 = \rho_1/\rho_2 \quad (11)$$

Note that p is independent of the reflectance function, illumination direction and intensity, and the surface normal of the two points. It is a photometric invariant that is easy to compute and does not vary with the position and orientation of the surface with respect to the sensor and the source. Further, it represents an intrinsic surface property that can be used for object recognition.

We have assumed that the scene is illuminated by a single light source. Now consider the same scene illuminated by several light sources. The brightness of any point can be written as:

$$I = \rho [k_1 R(s_1, v, n) + k_2 R(s_2, v, n) + \dots + k_n R(s_n, v, n)] \quad (12)$$

where s_1, s_2, \dots, s_n are the directions of the n sources that are visible to the surface point under consideration and k_1, k_2, \dots, k_n are proportional to the brightness of the n sources. Since the reflectance ratio is computed using neighboring points, it can be assumed that both points are illuminated by the same set of sources. Then, from (11) and (12) we see that the reflectance ratio p is unaffected by the presence of multiple light sources.

Note that by definition p is unbounded; if the second surface point is black, $I_2 = 0$, then $p = \infty$. From a computational perspective, this poses implementation problems. Hence, we use a different definition for p to make it a well-behaved function of the reflectance coefficients ρ_1 and ρ_2 :

$$p = (I_1 - I_2)/(I_1 + I_2) = (\rho_1 - \rho_2)/(\rho_1 + \rho_2) \quad (13)$$

Now, we have $-1 \leq p \leq 1$. We will use this definition of the reflectance ratio in the following sections.

3 Reflectance Ratio of a Region

To this point, we have focused on two neighboring points. Now consider a surface region that has constant reflectance coefficient ρ_1 and is surrounded by a background region with constant reflectance coefficient ρ_2 . We are interested in computing the reflectance ratio $P(S)$ of the surface region S with respect to its background. The brightness of the entire region cannot be assumed constant for following two reasons. (a) The surface may be curved and hence the surface normal can vary substantially over the region. (b) While the illumination may be assumed to be locally constant, it may vary over the region. These factors can cause brightness variations, or shading, over the region and its background as well. However, the reflectance ratio can be accurately estimated using neighboring (or nearby) points that lie on either side of the boundary between the region and the background. The reflectance ratio for a region can then be determined as an average of the reflectance ratios computed along the boundary of the region. The computed ratio is also a photometric invariant; it is independent of the three-dimensional shape of the surface and the illumination conditions.

Details of the reflectance ratio algorithm are given in [8]. Due to space limitations, we will simply outline the main steps of the algorithm. The algorithm can be divided in two parts. First, a sequential labeling algorithm [7] is used to segment the image into connected regions. During sequential labeling, the reflectance ratio of neighboring pixels is used as a measure of the "connectivity" between the pixels. In the second stage, a reflectance ratio for each segmented region is computed as the mean of the ratios computed for all points on the boundary of the region. The algorithm is computationally efficient in that reflectance ratios of all scene regions are computed in just two raster scans of the image [8].

4 Object Recognition

In this section, we apply reflectance ratios to the problem of object recognition. The recognition methods presented here are effective for objects that have markings with different reflectance coefficients. Man-made objects with pictures and text printed on them are good examples of such objects.

Learning Object Models:

Since, our objective is to recover the three-dimensional pose of an object from a single brightness image, the object model must include reflectance ratios of the object as well as the three-dimensional coordinates of the centroids of each region. This is done using a range finder. We use the image sensor of the range finder to also ob-

tain a brightness image of the object. As a result, the range and brightness images of the object are registered. The reflectance ratio algorithm is applied to the brightness image and the ratios (\hat{P}_m) and centroids (\hat{x}_m) (in the image) of the object's regions are determined. Next, the range map is used to obtain the three-dimensional coordinates (\hat{X}_m) of points of the object surface that correspond to the region centroids in the image. We assume that though the object surface may be curved, each constant reflectance region is small compared to the size of the object and hence can be assumed to be planar. Under this assumption, centroids of regions in the image correspond to centroids of the regions in the 3-D scene. Using the above approach, a ratio-centroid list $L_A = ((\hat{X}_1, \hat{P}_1), (\hat{X}_2, \hat{P}_2), \dots, (\hat{X}_m, \hat{P}_m), \dots)$ is obtained for each object. Here, $\hat{X}_m, m = 1, \dots, M$ are the 3-D centroids of the regions and $\hat{P}_m, m = 1, \dots, M$ are the reflectance ratios.

Next, a hash table [9] is initialized. All object models are stored in the same hash table. The indices in the hash table are invariants that can be computed from a single image of the scene. There are no useful geometric invariants that can be computed from the spatial arrangement of the region centroids [10]. This is because object rotation in the scene changes the relative configuration of the region centroids in the image. Hence, we rely on the photometric invariance of reflectance ratios for indexing into the hash table. We select three regions, i, j , and k on the object and use their reflectance ratios to obtain an index $\langle \hat{P}_i, \hat{P}_j, \hat{P}_k \rangle$. Indices are formed using only those region triplets (i, j, k) whose centroids in 3-D space lie within the radius of coherence D_A . This ensures that the number of indices generated is $O(N)$, with N the number of visible regions on the object, and not combinatorial in N . Associated with each index in the hash table is an entry. In the entry are stored, the object identifier \mathcal{M}_I , and the 3-D coordinates of the centroids ($\hat{X}_i, \hat{X}_j, \hat{X}_k$) of the three regions used in the index. The entry also includes the ratio-centroid pairs (\hat{X}_m, \hat{P}_m), of other object regions that are used for object verification and pose estimation.

The above procedure is applied to all sets of three regions in the list L_A . Each object is typically represented by several indices and entries in the hash table. This process is repeated for all objects, $\mathcal{M}_I, I = 1, \dots, \mathcal{O}$, of interest to the recognition system. The resulting hash table represents the complete object-model database which is ready for use by the recognition system.

Recognition and Pose Estimation

Though model acquisition requires the use of both a brightness and a range image of each object, recognition and pose estimation is accomplished using a single brightness image. The reflectance ratio algorithm is applied to the scene image to obtain the list $L_R =$

$((x_1, P_1), (x_2, P_2), \dots)$. For recognition, a set of three regions is selected from the list L_R . Consider the three regions (i, j, k) . This set is used only if the image centroids of the regions j and k lie within the radius of coherence D_R from the centroid of the region i . The ratios of the three regions are used to form the index $\langle P_i, P_j, P_k \rangle$. If this index does not have an entry in the hash table, the next set of three regions is selected from L_R . If an entry does exist, we have a hypothesis for the object (say \mathcal{M}_K). The entry includes the 3-D centroids of the regions (i, j, k) and a set of centroid-ratio pairs for other regions on the object \mathcal{M}_K . Assuming the object hypothesis is correct, we have a correspondence between the image centroids (x_i, x_j, x_k) and the 3-D centroids ($\hat{X}_i, \hat{X}_j, \hat{X}_k$) in the entry. Under the weak-perspective assumption, the transformation T from the 3-D scene points to 2-D image points can be computed from the three corresponding 3-D centroids and image centroids using the alignment technique proposed by Huttenlocher and Ullman [11]. In general, however, there exist two solutions to the transformation [11]:

$$x = T_{K1}(\hat{X}) \text{ and } x = T_{K2}(\hat{X}) \quad (14)$$

Weinshall [12] has shown that instead of computing these two transformations the inverse of the Grammian of the points \hat{X}_i, \hat{X}_j , and \hat{X}_k can be used to predict the image coordinates \hat{x}_o of a fourth 3-D point \hat{X}_o in the entry. Again, two solutions to \hat{x}_o exist but if the initial object hypothesis is correct, one of the two solutions is likely to be close to one of the centroids in the list L_R . Further, the reflectance ratio \hat{P}_o (in the entry) and P_o (in the list L_R) must be similar. The point \hat{x}_o is not guaranteed to be in the list L_R since it may not be visible to the sensor or it may be occluded by other objects in the scene. In any case, for the object to be verified, one or more projections of the 3-D regions in the entry must match in location and ratio with regions in the list L_R . If so, the object \mathcal{M}_K has been recognized and its pose is given by either T_{K1} or T_{K2} .

At this point, all regions used as indices and those that are verified are removed from the list L_R . A new set of three regions is selected from the list and used to form the next index. This process is repeated until either all objects in the hash table have been recognized or all regions in the list L_R have been explained.

5 Experiments

In this section, we present experimental results related to the invariance of reflectance ratios as well as the application of ratios to object recognition. Figure 2(a) shows a brightness image of an object with several constant reflectance regions. The image was obtained under ambient lighting conditions. Figure 2(b) shows the result obtained by applying the reflectance ratio algorithm. Ratio

values between -1.0 and 1.0 are offset and scaled to lie between 0 and 255 image brightness levels. Note that all letters in the word "KRYLON" have similar ratio values. The white dot in the center of each region represents the centroid of the region.

The invariance of computed ratios to the illumination direction is illustrated in Figure 3. The direction of a single light source is varied (about the axis of the object) from -70 degrees (left of the object) to 20 degrees (right of the object) in increments of 10 degrees. As seen from the figure, the reflectance ratio for region "K" demonstrates remarkable invariance to illumination direction.

The recognition experiments were conducted on man-made objects with letters and pictures printed on them. The printed regions have reflectance coefficients that depend on the shade or color of the paint used to print them. Figure 4(a) shows the model acquisition results for a 3-D object. The range image was obtained using a light stripe range finder. The vertices of the triangle displayed are the centroids of three regions whose reflectance ratios were used as indices in the hash table. Other nearby regions that are included in the hash table entry for object verification and pose estimation and their centroids are indicated by black boxes. All regions used for model acquisition and recognition are assumed to be planar though the objects they lie on may be curved. Under this assumption, the centroid of an image region corresponds to the projection of the centroid of the region in 3-D space. The planarity assumption is generally found to be reasonable for regions that are small compared to the size of the object.

Recognition and pose estimation is done using a single brightness image of the scene (see Figure 4(b)). The scene consists of several 3-D objects in different orientations and positions. It includes occlusions, shadows, and non-uniform illumination. The reflectance ratio algorithm was applied to the scene image and a total of 18 constant reflectance regions were detected. The index triangle shown in the model image is found and verified in the scene image. The set of three regions in the scene image produce a hypothesis for the object. Other regions in the object model are used to verify this hypothesis using the alignment technique [11]. The actual and projected centroids of the verification regions are indicated by black and white boxes, respectively. Some of the verification regions are not found in the scene image since they are occluded by other objects.

6 Conclusion

We conclude with the main points of this paper:

- We presented a photometric invariant, called region reflectance ratio, that is computed from a single brightness image.

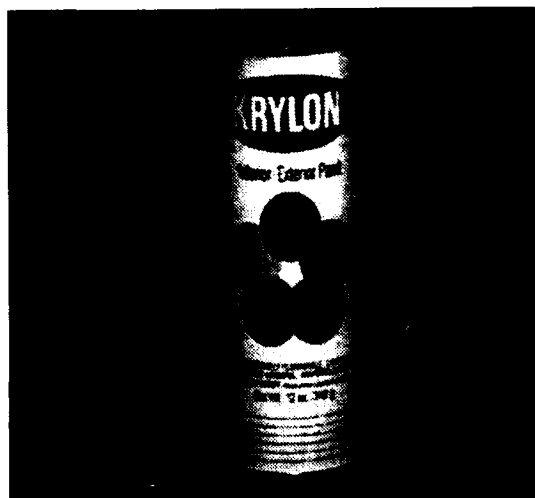
- We have used reflectance ratios to recognize objects from images. This approach is in contrast to previous recognition methods that rely solely on geometric features for recognition and pose estimation.
- The recognition technique presented here is capable of automatically learning models of the objects of interest.

Acknowledgement

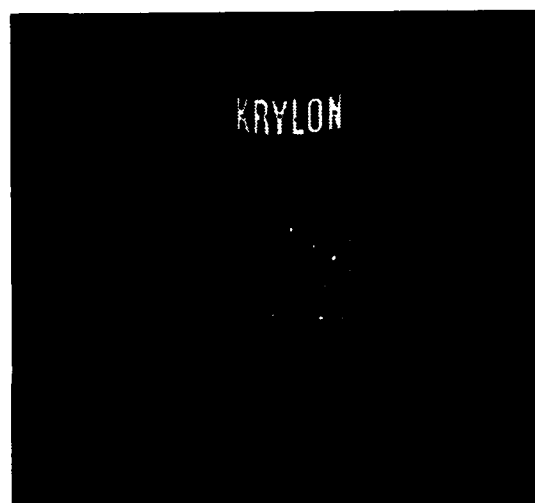
The authors would like to thank Ushir Shah for his assistance in implementing the model acquisition and recognition algorithms and Daphna Weinshall for providing the alignment code.

References

- [1] R. T. Chin and C. R. Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, 18(1), March 1986.
- [2] E. H. Land. The retinex. *American Scientist*, 52(2):247-264, June 1964.
- [3] E. H. Land and J. J. McCann. Lightness and retinex theory. *Journal of Optical Society of America*, 61(1):1-11, January 1971.
- [4] B. K. P. Horn. Determining lightness from an image. *Computer Graphics and Image Processing*, 3(1):277-299, December 1974.
- [5] G.D. Finlayson. *Colour Object Recognition*. M.S. thesis, Simon Fraser University, 1992.
- [6] S. K. Nayar, K. Ikeuchi, and T. Kanade. Surface reflection: Physical and geometrical perspectives. *IEEE Trans. on Pattern Analysis and Machine Intell.*, 13(7):611-634, July 1991.
- [7] B. K. P. Horn. *Robot Vision*. MIT Press, Cambridge, MA, 1986.
- [8] S. K. Nayar and R. M. Bolle. Reflectance based object recognition. Technical Report CUCS-055-92, Department of Computer Science, Columbia University, September 1992.
- [9] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Company, Reading, MA, 1974.
- [10] J.B. Burns, R. Weiss, and E. Riseman. View variation of point-set and line segment features. In *Proc. Image Understanding Workshop*, pages 650-659, April 1990.
- [11] D.P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *Int. Journal of Computer Vision*, 5(2):195-212, November 1990.
- [12] D. Weinshall. Model-based invariants for 3d vision. Technical Report RC 17705, IBM Thomas J. Watson Research Center, December 1991.



(a) Brightness image.



(b) Ratios and centroids of object regions.

Figure 2: Reflectance ratios of regions computed from a single image.

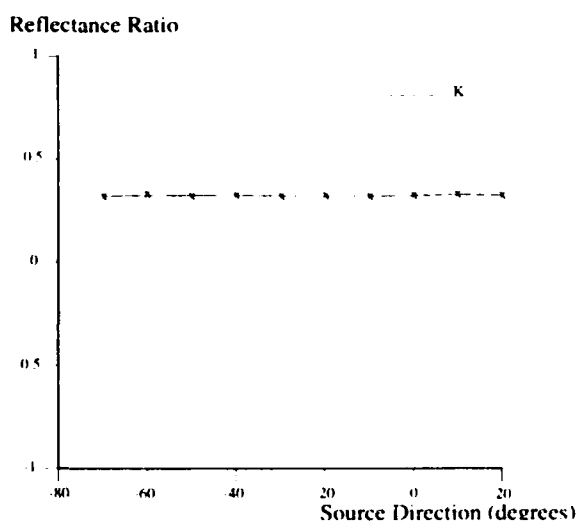
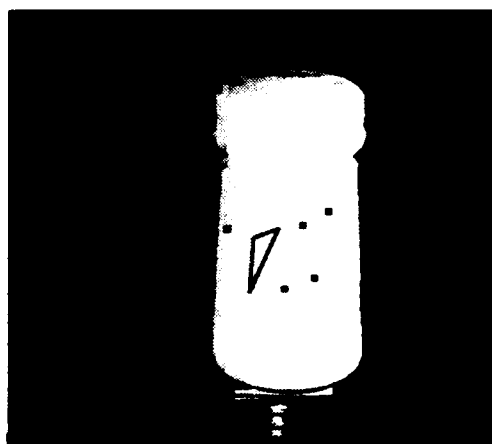


Figure 3: Invariance of reflectance ratios to the direction of illumination.



Brightness image.



Range image.

(a) Object model acquisition.



(b) Object recognition and pose estimation.

Figure 4: Model acquisition and object recognition results obtained for a three-dimensional recognition problem.

Robotic Three Dimensional Dual-Drive Hybrid Control for Tactile Sensing and Object Recognition*

Kelly A. Korzeniowski and William A. Wolovich

Laboratory for Engineering Man/Machine Systems
Division of Engineering,
Brown University,
Providence, RI 02912

1 Abstract

This work describes the development and testing of a 3-D dual-drive surface tracking controller that enables a robot to track along any specified path on the surface of an object. The dual-drive controller computes the normal and tangent vectors relative to movement along the path. The result is controlled movement in 3-D on the surface of an object. This tactile data collection method is referred to as "object-dependent" sensing because the location of the sensing paths is driven by comparisons made by the recognizer to a model data base. It is assumed that the path is generated by an external recognizer in such a way that the data points collected by tactile sensing along the path will maximize the probability of correctly identifying the object. The application for such a data collection system is object recognition tasks in environmental exploration and manipulation.

2 Overview

In the complete "object-dependent" tracking system, we envision an external recognition program that uses partial data sets collected by tactile sensors on the object's surface to attempt an identification and to direct future data collection to limit uncertainty in making an identification. "Object-dependent" tactile sensing with the 3-D dual-drive controller is an improvement over the general 3-D object tracking algorithm [Korz91] with the 2-D dual-drive controller. The 3-D dual-drive controller allows the robot to track along any path with the end effector in any orientation. Whereas the 2-D dual-drive controller limits tracking to the horizontal and vertical planes in the base frame. By adding recognizer information to the tracking scheme, tactile sensing is no longer limited to general tracking methods that may be spending time collecting extraneous data or ignoring features on an object that are critical to its identification.

*This work was partially funded by the NSF in conjunction with the Advanced Research Projects Agency of the Department of Defense under Contract No. IRI-8905436

Many applications of tactile sensing for data collection concentrate on driving the position of the robot by machine vision in order to collect discrete data points on the surface of an object. The work presented here differs in the respect that the dual-drive controller allows the robot to stay in contact with a surface and in general the result is that more data points can be collected in a shorter period of time. This paper will show that tactile force sensing can be used to control the robot and could be the main data collection method. Machine vision would be used as a secondary data collection method that supplements tactile surface tracking by providing information about the bounding outline, and an estimate for the surface normal and thus the approach vector for tactile surface contact points at the beginning of each path.

This controller is implemented and tested using an IBM 7565 Cartesian robot equipped with strain gauges on the end effector. The tracking controller is designed so that the robot will be able to track any free form complex object. Experimental results are presented to show that with the 3-D dual-drive controller, the robot is able to track an arbitrary path on a complex 3-D object.

3 Prior Results - Dual-Drive in 2-D

The 2-D dual-drive force/velocity controller is described in [Kaza88]. Given the value of the magnitude for the desired force and velocity, the action of the controller is to zero both the force and velocity errors. The result is that the robot maintains contact with an object, moving along the surface.

In this hybrid controller, separate servo loops handle the force and position calculations. The force servo loop is a damping controller with saturation. Basically, the force error, the difference in magnitude between the desired and actual force vectors, is converted to a velocity value by a damping constant. The resulting velocity is compared to the desired velocity and integrated to obtain a reference position for the robot.

The dual-drive controller assumes that the normal and tangent vectors will always be orthogonal. This relation between the velocity and the force can be

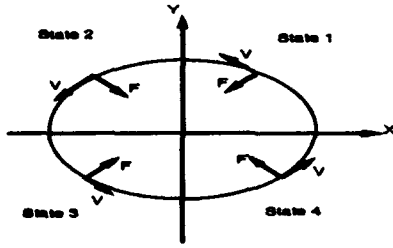


Figure 1: Determining the Sign of Force and Velocity

maintained if the tracking surface is sufficiently stiff and the friction between the surface and the end effector is negligible. If the surface is frictionless, the normal will be in the same direction as the force vector F (see Figure 1). Using the force sensor readings the outward unit normal and unit tangent vectors are given by,

$$N = \frac{1}{|F|}(F_x, F_y) \quad (1)$$

$$T = \frac{1}{|F|}(-F_y, F_x) \quad (2)$$

In the case of a stiff surface, the velocity due to force corrections is small and V is tangential. The outward unit normal and unit tangent vectors are,

$$N = \frac{1}{|V|}(V_y, -V_x) \quad (3)$$

$$T = \frac{1}{|V|}(V_x, V_y) \quad (4)$$

where $|F| = \sqrt{F_x^2 + F_y^2}$ and $|V| = \sqrt{V_x^2 + V_y^2}$. These four equations for the unit normal and unit tangent vectors give four different dual-drive implementations. See [Kaza88] for more details.

xsgn	+	state 1 or 4
	-	state 2 or 3
ysgn	+	state 1 or 2
	-	state 3 or 4

Table 1: Combinations for the Sign of Force and Velocity

In the tracking application considered here, the complete tracking trajectory, the shape of the object, is not known prior to tracking. As the robot moves around an object, the signs of the x-y components of the actual force and velocity change as shown in Figure 1. These changes are summarized in Table 1. The equations for determining the sign of the actual force and velocity vectors are,

$$\text{sgn}(V_{act}) = \text{sgn}[x\text{sgn}(V_y) - y\text{sgn}(V_x)] \quad (5)$$

$$\text{sgn}(F_{act}) = \text{sgn}[x\text{sgn}(F_x) - y\text{sgn}(F_y)] \quad (6)$$

The configuration of the dual-drive controller implied by the equations (1) through (6), was used to track and collect data points to identify 2-D objects. See [Brad90] for more details.

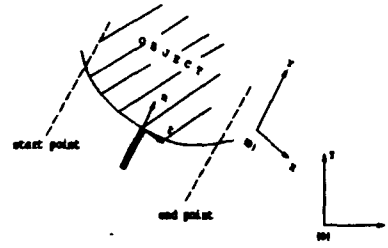


Figure 2: Top View - Cross Section of an Object

A general 3-D object tracking algorithm was developed using the 2-D dual-drive controller [Korz91]. The robot tracked around an unknown object, collecting horizontal planar slices of data for use in recognition. The subject of the remainder of this report is the expansion of the 2-D dual-drive control equations for use in tracking arbitrary paths in 3-D space.

4 General Surface Tracking Dual-Drive Control in 3-D

The dual-drive controller operates on the assumption that the actual normal and tangent vectors will be orthogonal. One of the requirements for this is that surface friction must be negligible. A low friction point for surface tracking occurs when the end effector is aligned near the surface normal. It has been found experimentally that this restriction can be relaxed so that the angle between the end effector and the surface normal, θ may actually be within a determined range, i.e. $|\theta| \leq \theta_{threshold}$. If the surface normal greatly varies along the tracking line and $|\theta| > \theta_{threshold}$, the end effector is reoriented, then the dual-drive equations are recalculated and tracking continues.

The extension from dual-drive control in 2-D to 3-D can be illustrated in the following way. A recognition program generates a bounded path that will be projected onto a surface. The path is projected by passing a plane between the bounds on the path generated by the recognizer. The orientation of the plane should be near the surface normal. This will become the orientation of the end effector, and the robot will track the path moving in the plane. Figure 2 illustrates the determination of the tracking path.

The direction of the vector connecting the bounded points of the path is the tangential direction of motion for the robot. As the surface changes locally, the normal and tangent vectors adjust and the result is that the robot's movement occurs in the 2-D plane. Dual-drive tracking occurs relative to the 2-D plane in the dual-drive D_a frame. This plane can be oriented in 3-space and given the appropriate mapping between the two frames the result is dual-drive tracking in the world frame O .

By using the orientation coordinate transforma-

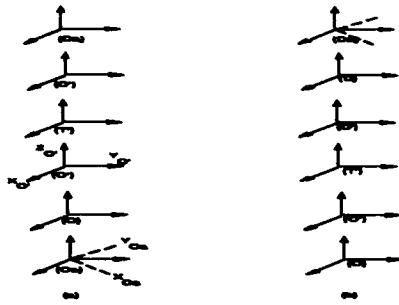


Figure 3: Frame Axes Between Base and Dual-Drive

tion matrices, for yaw pitch and roll rotations [Wolo87], to find the mapping for the force/velocity and position components between the O and D_a frames, $R_{D_a}^O$, one is able to make dual-drive 2-D force velocity calculations in reference to the directed tracking path. The components of force and velocity in the D_a frame are a function of the 3-D force components of the force and velocity in the O frame. The resulting 2-D motion in the dual-drive frame is mapped to 3-D motion in the base frame.

5 Mapping Vector Quantities Between Frame Axes

transformation matrix	function of angle
$R_{O'}^{O'}$	ψ_2
$R_{O_a}^{O'}$	ψ_1
$R_{O'}^{O_a}$	ψ_2
$R_{T'}^{O'}$	ρ
$R_{D'}^{T'}$	γ
$R_{D_a}^{D'}$	ψ_2
$R_{D_a}^{D'}$	ψ_1
$R_{D'}^{D_a}$	ψ_2

Table 2: Relating Orientation Matrices to Angles

For general yaw angle rotations ψ , the calculations for the mappings between frame axes is performed in two steps. Velocity readings are given by the system's sensors in reference to the O frame, $(v_{x_O}, v_{y_O}, v_{z_O})$. The force strain gauges are assigned to associated axes in frame O , $(f_{x_O}, f_{y_O}, f_{z_O})$. In Figure 3 (a), the x_{O_a} axis corresponds to the yaw of the end effector. The tool is oriented along the x_{O_a} axis. In order to make the necessary calculations, the velocities are needed in the O_a frame. First the x_{O_a} axis in frame O_a is yawed by an amount equal to $\psi_1 = \psi \bmod 90^\circ$ to rotate the axes to line up with the X, Y, or Z axis in the base frame O . This allows the force and velocity along the x_{O_a} and y_{O_a} axes to be expressed in terms of the forces and velocities in the x_O and y_O axes. The orientation rotation matrix used in this mapping is $R_{O_a}^{O'}$, a yaw matrix. The next yaw rotation, $\psi_2 = \psi - \psi \bmod 90^\circ$, is applied to frame O to align the axes in frame O'

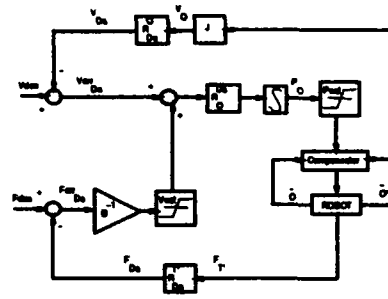


Figure 4: 3-D Dual-Drive Controller - Block Diagram

with the base frame. This is a necessary calculation in order to generalize the equations and correctly apply the following rotations. The rotation between the O' and T' frames, a pitch rotation ρ , brings the $x_{T'}$ axis in alignment with the surface normal. The rotation between the T' and D' frames, a roll rotation γ , brings the $y_{D'}$ axis in alignment with the vector that connects the bounding points on the tracking path. This is the direction of motion for the robot. A final yaw rotation of ψ_1 degrees is made between the D' and D_a frames. The dual-drive calculations in frame D_a are made with $F_{D_a} = f(f_{x_O}, f_{y_O}, f_{z_O}) = (f_{x_{D_a}}, f_{y_{D_a}})$ and $V_{D_a} = f(v_{x_O}, v_{y_O}, v_{z_O}) = (v_{x_{D_a}}, v_{y_{D_a}})$ along with the associated x_{sgn} and y_{sgn} for the vectors. The matrix equations become, $F_{D_a} = R_{D_a}^{D'} R_{T'}^{D'} R_{O'}^{D'} F_{O_a}$ and $V_{D_a} = R_{D_a}^{D'} R_{T'}^{D'} R_{O'}^{D'} V_{O_a}$. In Figure 3 (b), a similar calculations is made between the D and O frames for mapping the calculated displacements back to the base O frame. The yaw rotation that is applied first is ψ_1 then ψ_2 . The displacements are computed $P_{O_a} = f(p_{x_{D_a}}, p_{y_{D_a}}) = (p_{x_{O_a}}, p_{y_{O_a}}, p_{z_{O_a}})$. The matrix equation is $P_O = R_{O'}^{O'} R_{T'}^{O'} R_{D'}^{O'} P_{D_a}$. The matrices and the respective transposes are functions of the rotation angles as listed in Table 2.

The range on the total yaw angle rotation is, $0^\circ \leq \psi \leq 360^\circ$. This allows the robot to track around an object. The pitch rotation is related to the orientation of the end effector. The range on the pitch rotation is $-90^\circ \leq \rho \leq 90^\circ$. The roll rotation is related to the direction of motion given by the vector connecting the end points of the tracking path generated by the recognition program. The range on the roll rotations is $-90^\circ \leq \gamma \leq 90^\circ$. This range along with positive and negative velocity specifications for the robot's direction of motion, allows the robot to track along any given direction of motion.

The complete block diagram for the 3-D dual-drive controller is shown in Figure 4. The orientation mapping matrices are included in the diagram.

6 Matrix Multiplication

It was decided that the yaw and roll angles will be expressed as positive and the pitch angle will be expressed as negative. The matrices reflect this choice.

The force readings are taken in reference to the O_a frame according to the way the end effector is aligned along the axes. These force readings, $F_{O_a} = (f_{x_{O_a}}, f_{y_{O_a}}, f_{z_{O_a}})$ are mapped to the D_a frame, $F_{D_a} = (f_{x_{D_a}}, f_{y_{D_a}})$. Note that by definition, only the X and Y components of the forces in the D_a frame are used in the dual-drive calculations.

$$F_{D_a} = R_{D_a}^{D'} R_{D'}^{T'} R_{T'}^{O'} R_{O'}^{O_a} F_{O_a}$$

$$R_{D_a}^{O_a} =$$

$$\begin{pmatrix} c\psi_2 & -s\psi_2 & 0 \\ s\psi_2 & c\psi_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c\gamma & s\gamma \\ 0 & -s\gamma & c\gamma \end{pmatrix}$$

multiplied by

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c\psi_2 & s\psi_2 & 0 \\ -s\psi_2 & c\psi_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The resulting equations for F_{D_a} are,

$$f_{x_{D_a}} = (c^2\psi_2 + s^2\psi_2 c\gamma) f_{O_x} + (c\psi_2 s\psi_2 - c\psi_2 s\psi_2 c\gamma) f_{O_y} + (-s\psi_2 s\gamma) f_{O_z}$$

$$f_{y_{D_a}} = (s\psi_2 c\psi_2 - s\psi_2 c\gamma c\psi_2) f_{O_x} + (s^2\psi_2 + c^2\psi_2 c\gamma) f_{O_y} + (c\psi_2 s\gamma) f_{O_z}$$

The velocity readings are given by the sensors in reference to the O frame. These velocity readings $V_O = (v_{x_O}, v_{y_O}, v_{z_O})$ are mapped to the D_a frame, $V_{D_a} = (v_{x_{D_a}}, v_{y_{D_a}})$. In the D_a frame by definition there is no velocity along the z_{D_a} axis. The velocity equations to be used in the dual-drive calculations are given by the following matrix multiplications.

$$V_{D_a} = R_{D_a}^{D'} R_{D'}^{T'} R_{T'}^{O'} R_{O'}^{O_a} V_O$$

$$R_{D_a}^{O_a} =$$

$$\begin{pmatrix} c\psi_2 & -s\psi_2 & 0 \\ s\psi_2 & c\psi_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c\gamma & c\gamma \\ 0 & -s\gamma & c\gamma \end{pmatrix}$$

multiplied by

$$\begin{pmatrix} c\rho & 0 & -s\rho \\ 0 & 1 & 0 \\ s\rho & 0 & c\rho \end{pmatrix} \begin{pmatrix} c\psi_2 & s\psi_2 & 0 \\ -s\psi_2 & c\psi_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

multiplied by

$$\begin{pmatrix} c\psi_1 & s\psi_1 & 0 \\ -s\psi_1 & c\psi_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The resulting equations for V_{D_a} are,

$$v_{x_{D_a}} = v_{x_O} (c\psi_1 (c^2\psi_2 c\rho - s\psi_1 (s\gamma s\rho c\psi_2 - c\gamma s\psi_2)) - s\psi_1 (c\psi_2 c\rho s\psi_2 - s\psi_2 (s\gamma s\rho s\psi_2 + c\gamma c\psi_2))) + v_{y_O} (s\psi_1 (c^2\psi_2 c\rho - s\psi_2 (s\gamma s\rho c\psi_2 - c\gamma s\psi_2)) + c\psi_1 (c\psi_2 c\rho s\psi_2 - s\psi_2 (s\gamma s\rho s\psi_2 + c\gamma c\psi_2))) + v_{z_O} (-c\psi_2 s\rho - s\psi_2 s\gamma c\rho)$$

$$v_{y_{D_a}} = v_{x_O} (c\psi_1 (s\psi_2 c\rho c\psi_2 + c\psi_2 (s\gamma s\rho c\psi_2 - c\gamma s\psi_2)) - s\psi_1 (s\psi_2^2 c\rho + c\psi_2 (s\gamma s\rho s\psi_2 + c\gamma c\psi_2))) + v_{y_O} (s\psi_1 (c\psi_1 (s\psi_2 c\rho c\psi_2 + c\psi_2 (s\gamma s\rho c\psi_2 - c\gamma s\psi_2))) + c\psi_1 (s\psi_2^2 c\rho + c\psi_2 (s\gamma s\rho s\psi_2 + c\gamma c\psi_2))) + v_{z_O} (-s\psi_2 s\rho + c\psi_2 s\gamma c\rho)$$

The displacements for the robot must be specified in reference to the O frame. The dual-drive calculations produce the displacements $P_{D_a} = (p_{x_{D_a}}, p_{y_{D_a}})$. These are mapped to the O frame, $P_O = (p_{x_O}, p_{y_O}, p_{z_O})$.

$$P_O = R_{O_a}^{O'} R_{O'}^{T'} R_{T'}^{D'} R_{D'}^{D_a} P_{D_a}$$

$$R_{D_a}^{O_a} =$$

$$\begin{pmatrix} c\psi_2 & -s\psi_2 & 0 \\ s\psi_2 & c\psi_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c\rho & 0 & s\rho \\ 0 & 1 & 0 \\ -s\rho & 0 & c\rho \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c\gamma & -s\gamma \\ 0 & s\gamma & c\gamma \end{pmatrix}$$

multiplied by

$$\begin{pmatrix} c\psi_2 & s\psi_2 & 0 \\ -s\psi_2 & c\psi_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c\psi_1 & -s\psi_1 & 0 \\ s\psi_1 & c\psi_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The resulting equations for P_{O_a} are

$$p_{x_{O_a}} = p_{x_{D_a}} (c\psi_1 (c\rho c^2\psi_2 - s\psi_2 (s\gamma s\rho c\psi_2 - c\gamma s\psi_2)) + s\psi_1 (c\rho c\psi_2 s\psi_2 + c\psi_2 (s\gamma s\rho c\psi_2 - c\gamma s\psi_2))) + p_{y_{D_a}} (-s\psi_1 (c\rho c^2\psi_2 - s\psi_2 (s\gamma s\rho c\psi_2 - c\gamma s\psi_2)) + c\psi_1 (c\rho c\psi_2 s\psi_2 + c\psi_2 (s\gamma s\rho c\psi_2 - c\gamma s\psi_2)))$$

$$p_{y_{O_a}} = p_{x_{D_a}} (c\psi_1 (c\rho s\psi_2 c\psi_2 - s\psi_2 (s\gamma s\rho s\psi_2 + c\gamma c\psi_2)) + s\psi_1 (c\rho s^2\psi_2 + c\psi_2 (s\gamma s\rho s\psi_2 + c\gamma c\psi_2))) + p_{y_{D_a}} (-s\psi_1 (c\rho s\psi_2 c\psi_2 - s\psi_2 (s\gamma s\rho s\psi_2 + c\gamma c\psi_2)) + c\psi_1 (c\rho s^2\psi_2 + c\psi_2 (s\gamma s\rho s\psi_2 + c\gamma c\psi_2)))$$

$$p_{z_{O_a}} = p_{x_{D_a}} (c\psi_1 (-s\rho c\psi_2 - s\psi_2 s\gamma c\rho) + s\psi_1 (-s\rho s\psi_2 + s\gamma c\rho c\psi_2)) + p_{y_{D_a}} (-s\psi_1 (-s\rho c\psi_2 - s\psi_2 s\gamma c\rho) + c\psi_1 (-s\rho s\psi_2 + s\gamma c\rho c\psi_2))$$

7 Experiments and Applications

The 3-D dual-drive controller was implemented and tested using an IBM 7565 Cartesian robot. Presently, the actions of the recognition system are simulated. The location and bounds on the tracking path and the orientation of the end effector are supplied manually. Given the end effector orientation and the approach vector, the robot moves toward the surface until contact is made as indicated by the force sensing strain gauges. At this point surface tracking and data collection commences.

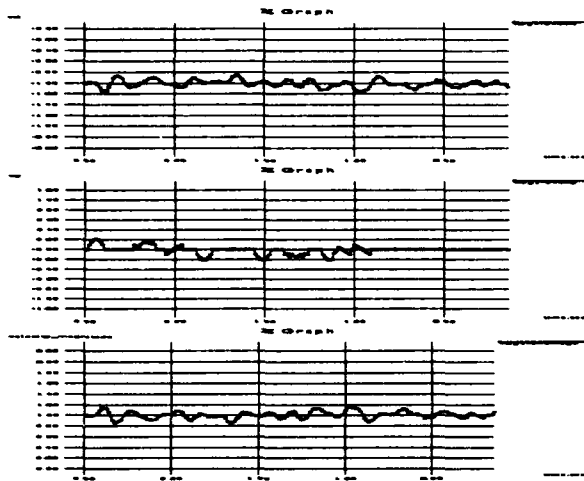


Figure 5: Velocity Data Plots, $|V_{desired}| = 1.0 \text{ in/sec}$

Controller modifications are added to monitor θ and the force at the end effector. If surface contact is lost during tracking the dual-drive controller is temporarily deactivated until contact is regained via a surface searching routine. The robot moves in an outward turing square spiral until contact is regained, then surface following continues.

A reading lamp was chosen to show the performance of the 3-D dual-drive controller. In Figure 5 and Figure 6 the velocity components $v_{x_{D_a}}$, $v_{y_{D_a}}$, $|v_{D_a}|$ and force components $f_{x_{D_a}}$, $f_{y_{D_a}}$, $|f_{D_a}|$ relative to the D_a frame are shown in the following graphs. This data was collected as the robot moved along a typical tracking path. In the force magnitude $|f_{D_a}|$ and the velocity magnitude $|v_{D_a}|$ graphs, the magnitude of the desired force and velocity is indicated on each of the graphs. Again the performance of this controller is comparable to the 2-D case as reported in [Kaza88].

8 Conclusions and Future Work

This report has explained the development of a 3-D dual-drive controller. The application for this controller is tracking line segments on an object's surface in order to recognize the object. It is assumed that an external recognition program generates the location of the tracking line. Data points that are collected as the robot moves along the line will limit the uncertainty with which an identification can be made.

Given a desired force and velocity, the dual-drive controller zeroes both force and velocity errors so that the robot's end effector moves across the surface. The result is 3-D controlled tracking motion. It has been demonstrated that the controller can be successfully applied to tracking real-world objects.

Work is currently underway to develop a 3-D tracking algorithm. The chief purpose of the algorithm will be to combine continuous and discontinu-

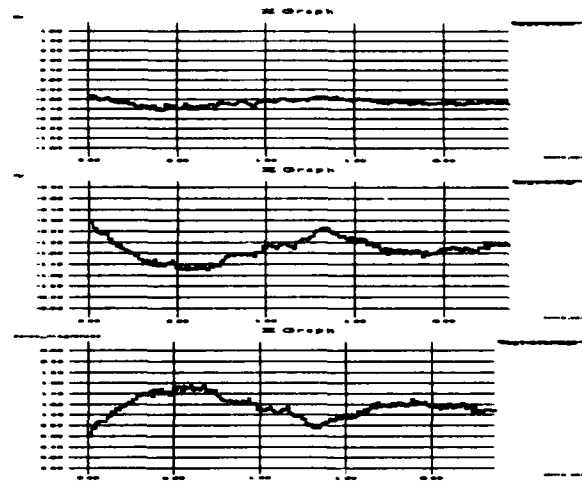


Figure 6: Force Data Plots, $|F_{desired}| = 1.1 \text{ lbs}$

ous tracking methods in order to more intelligently apply the controllers to handle changes in surface along the tracking path interval. Prior knowledge about surfaces that appear in the environment may also be included depending on the requirements of the tracking algorithm and object recognition. The purpose of this work will be to develop an algorithm that will enable the robot to track complex real-world objects on a specified tracking interval using the 3-D dual-drive controller.

References

- [Alle87] Allen, P.K., "Robotic Object Recognition Using Vision and Touch", Kluwer Academic Publishers, Norwell, MA, 1987.
- [Alle89] Allen, P.K., Roberts, K.S., "Haptic Object Recognition Using a Multi-Fingered Dextrous Hand", *IEEE Conference on Robotics and Automation*, Scottsdale, AZ, May 1989.
- [Brad90] Bradley, N.S., "Force Control Methods for Environment Interaction and Recognition", *Master's Thesis, Brown University*, May 1990.
- [Kaza88] Kazanzides, P., "Multiprocessor Control of Robotic Manipulators", *Ph.D. Thesis, Brown University Technical Report LEMS-47*, May 1988.
- [Khat87] Khatib, O., "A Unified Approach for Motion and Force Control of Robotic Manipulators: The Operational Space Formulation", *IEEE Journal of Robotics and Automation*, vol. 3, no. 1, 1987, pp. 43-53.
- [Korz91] Korzeniowski, K.A. and Wolovich, W.A., "Robotic 3-D Object Recognition Using Dual-Drive Control", *Proceedings of the 30th IEEE Conference on Decision and Control*, December 1991.
- [Rayf88] Rayfield, J.T., "Armstrong, a Loosely-Coupled Multiprocessor Testbed for Reconfigurable Topologies", *Ph.D. Thesis, Brown University*, May 1988.
- [Wolo87] Wolovich, W.A., "Robotics: Basic Analysis and Design", Holt, Rinehart, and Winston, 1987.

Section XIV

Segmentation and Partitioning

*Inferring Global Perceptual Contours from Local Features**

Gideon Guy and Gérard Medioni

Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273

Abstract

We attempt to solve the problem of imperfect data produced by state-of-the-art edge detectors through the implementation of laws of *Perceptual Grouping*, derived from the psychology field.

We introduce a saliency-enhancing operator capable of highlighting features (edges, junctions etc.) which are considered 'important' psychologically. It also infers features which are not detected by low-level detectors. We show how to extract salient curves and junctions and generate a description ranking these features by the likelihood of them occurring accidentally.

We also treat the problem of illusory contours apparent in end-point formations. The scheme is particularly useful as a gap filler and in the presence of a large amount of noise. It is interesting to note that all operations are parameter-free, non-iterative and are linear with the number of edges in the input image.

1 Introduction

An area which is likely to improve results in computer vision is the one of perceptual grouping. Perceptual Grouping can be classified as a mid-level process directed toward closing the gap between what is produced by state-of-the-art low-level algorithms (such as edge detectors) and what is desired as input to high level algorithms (perfect contours, no noise, no fragmentation, etc.). Many researchers resort to using synthetic data as their input because of these weaknesses.

Methods for edge labeling (like [Clowes 1971], [Waltz 1975]) assume perfect segmentation and connectivity, and define constraints which are only valid under these assumptions. These methods cannot work on 'real' edges. Other methods, like shape from contour [Ulupinar 1991], and representation of objects ([Rom

and Medioni 1993], [Sugihara 1984]), also rely heavily on the connectedness of the edges, and can benefit from the removal of noise (erroneous segments). Pattern recognition schemes (like [Stein and Medioni 1992]) rely on (at least) partial connectedness of the edges, and cannot function if the edge image is very fragmented. Also, the amount of noise is directly proportional to the computational cost of finding 'real' objects in a scene.

Using global perceptual considerations when attempting to connect fragmented edge images can alleviate many of the above problems, as shown later.

In a previous paper ([Guy and Medioni 1992a]), we have introduced a general algorithm capable of highlighting features due to co-curvilinearity and proximity. We suggested the **Extension Field** as a 'voting pattern' representing a large family of smooth curves, all at once. Here we further explore the properties of the Extension Field. We also suggest specialized fields that can be used within the same computational paradigm to reveal perceptual phenomena such as end-point formations and straight lines. Experimental results with different types of distortion applied to the input are also presented. We start by explaining the original algorithm, then describe some of the properties of the Extension Field. We follow by a comparison with the classical Hough transform and conclude by offering a set of other fields for special purposes.

1.1 Perceptual Grouping

Perceptual Grouping refers to a class of visual phenomena where clustering of physically non-connected elements in the image occurs. This task is equivalent to a figure-ground discrimination when patterns are embedded in noise. Figure 1 depicts examples of perceptual groupings which are of interest to us, and considered to be the result of a pre-attentive process. Such processes are known to take several hundreds of milliseconds (200-500 ms) to complete, and are thus not likely to utilize any high-level reasoning mechanism in the brain [Boff, et al. 1986].

The circle in the middle of figure 1(a) is easily distinguishable from its noisy background. Furthermore,

* This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-90-C-0078. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

we tend to fill the gaps, or more precisely, we are able to complete the circle mentally. The same holds for the geometrical patterns in 1(b). Figure 1(c) depicts a dot formation. Again, grouping of certain dots is possible, and salient curves are noticeable. A more striking example of *illusory* contours is found in the Kanizsa illusion [Kanizsa 1976] shown in Figure 1(d). Here we perceive edges which have no physical support whatsoever in the original signal.

The Gestalt psychologists (e.g. [Boff, et al. 1986]) were among the first to address the issues of pre-attentive perception. Many 'laws of grouping' were formulated, but none put in any computational (or algorithmic) language. Furthermore, the rules tend to supply conflicting explanations to many stimuli. This makes the computational implementation of such laws non-trivial. With input in the form of edges, the laws most relevant to our work relate to *proximity* and *good continuation*.

Lowe [Lowe 1987] discusses the Gestalt notions of *co-linearity*, *co-curvilinearity* and *simplicity* as important in perceptual grouping. Ahuja and Tuceryan [Ahuja and Tuceryan 1989] suggest methods for clustering and grouping sets of points having an underlying perceptual pattern.

Dolan and Weiss [Dolan and Weiss 1989] demonstrate a hierarchical approach to grouping relying on compatibility measures such as proximity and good continuation. Mohan and Nevatia [Mohan and Nevatia

1989a] assume a-priori knowledge of the contents of the scene (i.e. aerial images). A model of the desired features is then defined, and groupings are performed according to that model. In a later work [Mohan and Nevatia 1989b], groupings based explicitly on symmetries are suggested, but the first connectivity steps are performed locally.

Sha'ashua and Ullman [Sha'ashua and Ullman 1988] suggest the use of a saliency measure to guide the grouping process, and to eliminate erroneous features in the image. The scheme prefers long curves with low total curvature, and does so by using an incremental optimization scheme (similar to dynamic programming).

The main features of some of the more important works are summarized in table 1, and contrasted with our scheme. It is interesting to note that virtually all proposed algorithms use *local* operators to infer more global structures. Also note that many of the schemes are iterative, relying on one relaxation (or minimization) scheme or another, and are similar in that sense. The main differences are in the choice of the compatibility measures or the function to minimize.

2 Overview of Our Approach

As was demonstrated before, the physical evidence extracted *locally* from images (e.g. through edge detectors) is in many cases ambiguous and does not fully correspond to the human perception of the image. It is thus necessary, we believe, to impose *global* perceptual considerations at the low-level process.

In our method, *each* site (pixel or other cell) collects votes from *every* segment in the image. These votes contain orientation and strength information preferred by the voting segment. A measure of 'agreement' (in terms of orientation) is now computed, and sites which have high agreement values are considered salient. In more technical terms, a vector field¹ is generated by each segment, and a function over the whole space determines points of saliency. A subsequent step links areas of high saliency to produce a description in terms of curves and junctions.

Our voting scheme is somewhat related to the Hough transform approach [Hough 1962], but can detect shapes defined by their *properties* (smoothness etc.) rather than by their *exact shape* (lines, circles, etc.). A study of the relations between our scheme and the Hough transform is given in section 11.

The process is likely to produce features more similar to what we perceive, both in terms of saliency and connectivity. Also, since noise is not likely to produce high 'agreement' values, it becomes attenuated and thus reduces the complexity of the image (e.g. in terms of the number of useful edges).

1. Which we later call the Extension Field.

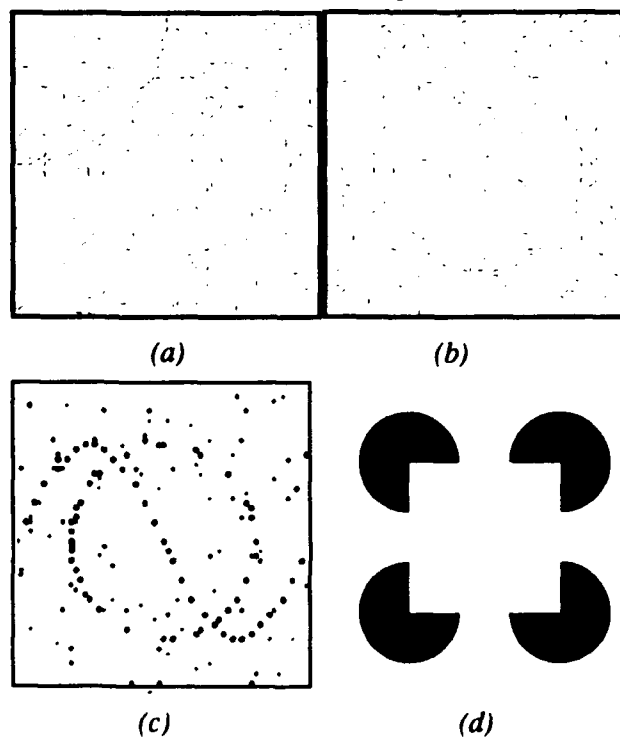


Figure 1 (a) & (b) Two instances of perceptual arrangements. (c) A dot formation. (d) The Kanizsa Square.

Table 1: Comparison of different grouping techniques

	[Lowe 1987]	[Ahuja and Tuceryan 1989]	[Dolan and Weiss 1989]	[Mohan and Nevatia 1989b]	[Sha'ashua and Ullman 1988]	Our scheme
Operator	Local	Local	Local	Local	Extensible (local)	Global
Primitives	Straight lines	Dots	Straights	curves	straight lines and curves	dots, lines and curves
Control	One pass	Iterative	Iterative	Relaxation Progressive	Parallel-progressive (iterative)	One-pass convolution
Noise immunity	Not clear	Good	Good	Good	Moderate	Very good
Scale	One	One	Hierarchy	One	One	One
Parameters	Yes	No	Yes	Yes	Yes	None
Pre-attentive (Domain free)	Yes	Yes	Yes	No (yes)	Yes	Yes
Special feature	First	Dot clustering, parameter free	Multi-resolution	Symmetry, high-level con.	Saliency map	Saliency map, unified, parameter-free
Sensitive computations	Yes	No	Yes?	Yes	Yes	None

2.1 Model of the Input

We would like to associate with each site of an image a direction, strength and a degree of uncertainty for that direction. This can be nicely approximated by an ellipse

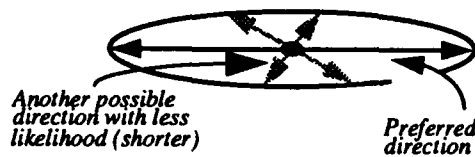


Figure 2 A simple input site model. Every site is associated with a preferred direction, strength and eccentricity (or uncertainty).

ellipse (as illustrated in figure 2). With such an input model, one site could be classified as being a part of a curve with known orientation and no uncertainty, while another as being a *point* with completely uncertain orientation.

We can thus use as input either a thresholded output of any edge detector (with no linking) or even an unthresholded version of the edge detector output. It can be verified experimentally that our system yields almost the same results with different choices of this threshold, as long as a sufficient number of useful features are present [Guy and Medioni 1992b].

Uncertainty with respect to orientation is inherent in the process of edge detection since, in many cases, a discrete set of oriented masks are convolved with the image, and the one with the largest response determines

the direction of the local edge. This value is accurate only up to the resolution of the set of masks, and can be used as an uncertainty measure in our scheme.

Obviously, our ability to extract useful features is directly proportional to certainty, because the information content is reduced when uncertainty is introduced in the input.

2.2 Model of the Output

Our model of the output is related to Ullman and Sha'ashua's [Sha'ashua and Ullman 1988] in the sense that a *saliency map* is first constructed from an edge image, and higher-level features are inferred later. The saliency map assigns a value and a direction to *every* position in the image.

Ideally, such saliency map should assign large values of likelihood along illusory lines (as well as along physical curves), and also specify a direction of most probable continuation of any given segment. This will enable us, at a later stage, to group features by following the salient connections between the primitives. The map should assign a value of zero to areas of no saliency. Furthermore, curves with strong saliency should be assigned larger values than weaker curves.

2.3 Rationale for the Extension Field

In order to define saliency qualitatively, we start by writing down the major constraints which govern our mechanisms of saliency (at least according to the theory of the Gestalt).

2.3.1 The perceptual constraints

Our underlying goal is to keep the interpretation as simple as possible in the 'Gestalt' sense. This translates into four major constraints:

- 1) Co-curvilinearity - In the lack of other cues, smooth continuation is the only interpretation, and so is co-curvilinearity.
- 2) Constancy of curvature - We tend to extend a curve of some constant curvature with the same curvature, keeping the interpretation as simple and regular as possible, yet consistent with our sensory information. This principle is called *Prägnanz* by Gestaltists (see Figure 3).

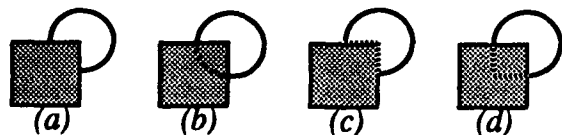


Figure 3 An obscured figure (a) triggers the perception of simple shapes (b), instead of the more complex (c) and (d).

- 3) Favoring low curvatures over large ones - Humans seem to connect fragmented line segments in a way that the increase in total curvature is minimum (see [Sha'ashua and Ullman 1988]).
- 4) Proximity - Closer segments influence each other more than distant ones.

With that in mind, we have devised a technique that implicitly imposes the above constraints in the form of an Extension Field emanating from each edge segment, as discussed in the next few sub-sections.

2.3.2 Extending a Curve

Given a line segment we ask the question: What is the shape of the most 'natural' extension, based on the mentioned constraints?

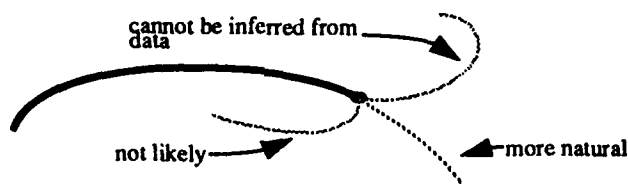


Figure 4 What is the shape of the most 'natural' extension to a given curve?

Many approaches in the past (e.g. [Dolan and Weiss 1989], [Lowe 1987]) used the *tangent* of the end-point to determine the best extension. This approach cannot always work properly for three reasons:

- 1) The tangent is very sensitive to noise and may introduce large errors.
- 2) The end-point may not be determined uniquely if the curve in question is fragmented.
- 3) The extension can only be a straight line (first order), thus not taking into consideration the global shape of the curve.

We would thus want to consider *all* the curve in such a way that the extension is smooth, influenced most by the behavior of close-by points of the curve, but can assume any form. Note that such extension can be constructed even if the curve is fragmented, and is robust, being the 'average' of many segments.

We go further than that. Rather than having only the best extension, we would like to list *all* possible extensions in the order of their likelihood. This suggests the use of some kind of a *field* (or flow) emanating from the end-point of a segment. The idea of a *field* plays a major role in our scheme, as we will show later.

2.3.3 Best Connection between Two Line Segments

The situation occurs whenever a 'compatibility figure²' of two close-by segments is computed in order to decide whether these segments should be grouped together. When determining the compatibility of two lines we would like to consider for each line its best extension (or extension field, as discussed in 3.1) and arrive at some compromise as to the best path between them. In



Figure 5 What is the best path between these two curves?

other words, each of the curves votes (using its field) for a family of curves. If the curves should really be connected then some extension from curve 1 would align with another extension of curve 2, to form the *compromise*³. The idea of a *compromise* between extension fields is also central to the approach, and will be presented in a more formal way later.

It is not reasonable to expect that extension curves from two different extension fields will align throughout their extent. It is more likely that such extensions align locally in many places. For that reason, the extension field will consist of local best candidates for extensions. In the next section we define the exact shape and usage of the Extension Field.

3 Extension, Point, and intermediary Fields

3.1 The Extension Field

Definition: An *Extension Field* is a non-normalized probability directional vector field describing the contribution of a single unit-length edge element to its environment in term of length and direction.

In other words, it votes on the preferred direction

2. A measure of 'agreement' between two curves, based on the difference of the end-point tangents and separation. (as used by [Dolan and Weiss 1989], [Lowe 1987], [Mohan and Nevatia 1989a])

3. Which is not necessarily the *best* continuation of any of the existing curves, or even a connection between the given end-points.

and the likelihood of existence of every point in space to share a curve with the original segment. The field is of *infinite* extent, although in practice it disappears at a predefined distance from the edge. Figure 1 depicts the Extension Field.

3.2 Design of the Extension Field (Orientation and strength)

Since we favor small and constant curvature, field direction at a given point in space is chosen to be tangent to the osculating circle passing through the edge segment and that point, while its strength is proportional

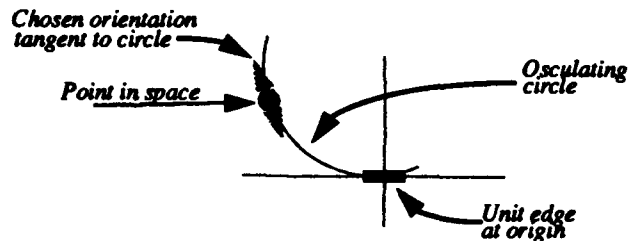


Figure 6 Assigning a direction for every point in space

to the radius of that circle. Also, the strength decays with the distance from the origin (the edge segment). The choice of a circular extension agrees with the constraint of smallest total curvature.

In trying to computationally evaluate the various constraints over a given curve we find that a (somewhat

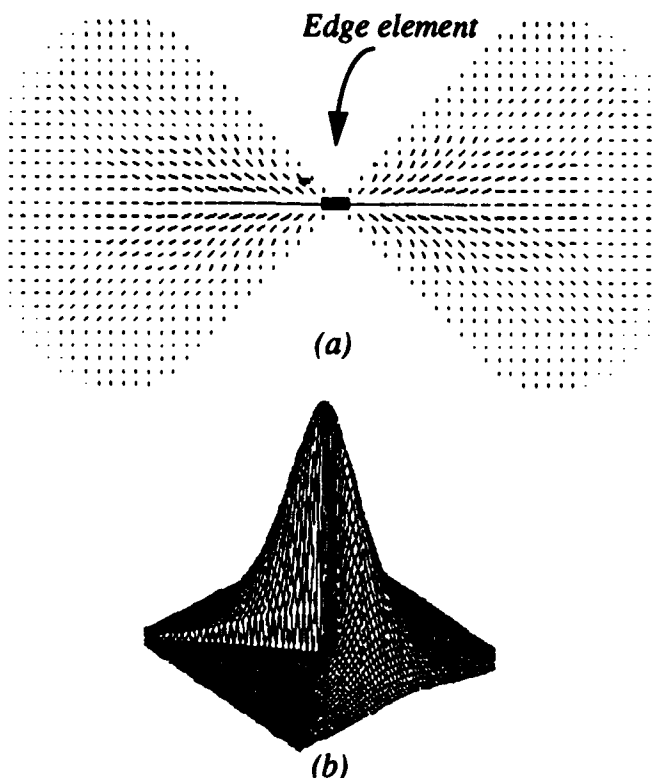


Figure 1 The basic Extension Field. (a) Direction, and (b) Strength.

revised⁴) measure of Total Curvature (as used by [Sha'ashua and Ullman 1988]) encompasses most of the desired constraints. We define the Total Curvature (TC) to be:

$$TC = \int \left| \frac{d\theta}{ds} \right|^\alpha ds, \quad \alpha > 1 \quad (1)$$

This is the integral of the curvature (brought to some power) along the curve, where θ is the tangent along the curve γ . The variable α is traditionally taken to be equal to 2, but it can be shown that the choice of a circle as the connecting curve in the scenario shown in Figure 6 minimizes the TC for all values of α greater than 2 [Guy and Medioni 1992b]. A larger α would just penalize sharp turns more than a smaller one. We thus set the orientations of the field elements to be tangent to a circular arc connecting the origin and that point in space.

Although different grouping laws compete, we claim that by finding the correct weighting values once, and given a sufficient amount of useful data⁵, we can resolve most conflicts. The best way to determine these values is by considering an *intentionally* ambiguous or *undecidable* case. The assignment of actual probabilities to the field is thus performed as follows: We consider two short edge segments, *perpendicular* to each other and apart⁶ (see center of Figure 7). This sce-



Figure 7 5 arrangements of two segments, each with a different separation angle. Angles much smaller than 90 degrees suggest a corner, while angles much larger than 90 degrees suggest a smooth connection.

nario, we claim, is a middle point between a clear choice of a connection by a sharp junction and a connection by a smooth curve. We regard this to be a most competitive scenario in terms of grouping of the two line segments. We thus assign probabilities to the field elements in such a way that all paths connecting these segments are assigned roughly the *same* saliency, and there *does not* exist any single best path between the two. More precisely, we set the field element strengths such that all values within the marked triangular region are the same. Such

4. In [Sha'ashua and Ullman 1988], $\alpha=2$ is used, and the absolute value is redundant.

5. A more precise definition of 'sufficient' is given in section 10.5, while addressing the issue of noise.

6. This scenario is termed 'the maximum undecidability arrangement'.

a scenario, when repeated for all scales, removes all degrees of freedom as to the choice of values for the field.

The reason all values beyond the two main diagonals are zeros, is a technical one. Having a segment vote for a point in space which is more than 90 degrees away (along a circle) could potentially cause unrelated segments to vote for the same curve, even though such a curve should not connect them. Also, it is quite obvious that extending a curve beyond the 90 degree point does *not* satisfy the minimum curvature constraint.

3.3 The Point Field

A dot image, is a degenerated case of an edge image, where the edges have no direction. Such maximum uncertainty⁷ in the input fits our input model well, and allows us to handle such cases in a uniform way. Obviously, perception is weakened by the loss of orientation data, and we are only able to handle cases with a moderate amount of noise. The only applicable perceptual law is that of *proximity*.

A suitable field must have circular symmetry, and in practice is constructed by convolving our original extension field with a 'multi-directional' edge segment. A typical input is shown in Figure 1(c), where a broken sine wave and a random set of points on a circle are embedded in noise.

Another scenario where the point field could be useful is in images where co-curvilinear formations between features other than dots are present. In such cases we would like to treat the image as if it was made out of non-directional tokens (or dots), and apply the point field to it.

We unify the maximum certainty (Extension) field and the Point field (and all fields in between) by considering a continuum of eccentricities associated with the multi-directional edge. The Extension field is constructed from a degenerated instance of an ellipse (a line), while the Point field is created from a 'circular' edge point. This makes full use of our input model (in section 2.1) and allows treatment of mixed images⁸ in a consistent and unified way.

4 Computation of the Saliency Map

4.1 Directional Convolution

The process of computing the saliency map can be thought of as a directional convolution with the above field (mask). The resulting map is then a function of a collection of fields, each oriented along a corresponding short segment. Each site accumulates the 'votes' for its own preferred direction and strength from every other site in the image. These values are combined at a site as described next. When long curves are present in the in-

put, they are 'cut' into unit length segments and each convolved with the field. We will discuss (in section 6) the properties of the resulting field when applied to a long curve (either straight or curved). The strength of the field is proportional to the strength⁹ of the segment, so that stronger segments have stronger votes, throughout the space.

Note that, although the process is local in essence, the fields impose some global order, and one line segment can implicitly 'vote' for a large curve without any *explicit* global reasoning involved.

4.2 Combination at each Site

Ideally, we would want an averaged majority vote regarding the preferred orientation of a given position. In practice, we treat the contributions to a site as being vector weights, and compute moments of the resulting system. Such a physical model behaves in the desired way, giving both the preferred direction and some measure of the agreement. We use the direction of the principal axis (*EVmin*) of that physical model as the chosen orientation (See equation (2)).

$$\begin{bmatrix} m_{20} & m_{11} \\ m_{11} & m_{02} \end{bmatrix} = \begin{bmatrix} EVmin \\ EVmax \end{bmatrix} \begin{bmatrix} \lambda_{min} & 0 \\ 0 & \lambda_{max} \end{bmatrix} \begin{bmatrix} EVmin^T & EVmax^T \end{bmatrix} \quad (2)$$

This acts as an approximation to the desired majority vote, without the need to consider the individual votes, but rather the statistics of the ensemble.

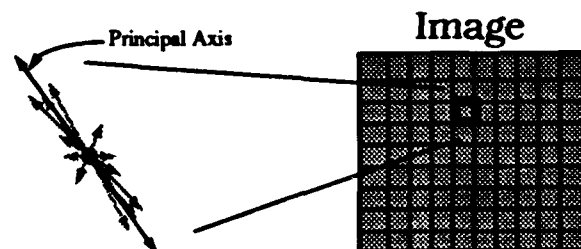


Figure 8 The principal axis of the votes collected at a site is taken as an approximation of the preferred direction.

The saliency map *strength* values are taken as the values of the corresponding λ_{max} at each site. So, large values would indicate that a curve is likely to pass through this point. This map can be further enhanced (as shown later) by considering the eccentricity, or $1 - (\lambda_{min}/\lambda_{max})$. When that value is multiplied by the previous saliency map we achieve better selectivity, and only *curves* are highlighted. This results in a map defined by $\lambda_{max} - \lambda_{min}$.

4.3 Justification

Basically, what we are looking for is a function that accepts positive vectors as input and results in a measure of the agreement in their orientation. The result should satisfy several criteria:

7. w.r.t. orientation

8. i.e. having varying certainty measures (or a mixture of dots and lines)

9. in terms of contrast.

- We want the result to be normalizable, so that we can compare different sites on a standard scale.
- The measure needs to be monotonically increasing with the addition of positive contributions.
- It should give higher values to 'better' (more directed) spatial arrangements of vectors.
- We want the affect of proximity to be independent to the affect of agreement.

It is easy to show how the model behaves when a single vector is added to it. Assume the variance-covariance matrix is as follows at state t :

$$C^t = \begin{bmatrix} m_{20}^t & m_{11}^t \\ m_{11}^t & m_{02}^t \end{bmatrix} \quad (3)$$

The sum of the eigenvalues is the trace of the matrix:

$$\lambda_{\min}^t + \lambda_{\max}^t = m_{20}^t + m_{02}^t \quad (4)$$

Now adding a new vector $V = [R \cos \theta, R \sin \theta]^T$ to the system will result in a new state $t+1$:

$$\lambda_{\min}^{t+1} + \lambda_{\max}^{t+1} = m_{20}^t + m_{02}^t + (R \cos \theta)^2 + (R \sin \theta)^2 = m_{20}^t + m_{02}^t + R^2 \quad (5)$$

Note that the angle θ has disappeared on the r.h.s. of (5). This means that the sum of eigenvalues is independent of the orientations of the voting vectors and can hence be used as an indicator of proximity (a wider sense of proximity of course), and as a primitive saliency measure.

Equation (5) can obviously be written as:

$$\lambda_{\min} + \lambda_{\max} = \sum_{i=1}^N R_i^2 \quad (6)$$

Where N is the number of segments in the original image.

We define the eccentricity $E = 1 - \lambda_{\min} / \lambda_{\max}$ as a measure of agreement. Obviously this value is between 0 and 1¹⁰. Our intuitive notion of 'agreement', or of a majority vote on a continuous scale, is consistent with the above definition. This means that in all cases where we feel that collection A has better 'agreement' than collection B , the corresponding eccentricity values will share the same relationship (i.e. $E(A) > E(B)$). This is not to say that both functions are equal, but merely that both are monotonic.

Eccentricity values by themselves cannot perform as saliency measures since sites with very little voting strength can produce high eccentricity values. In fact, consider a site far away from where the 'action' is, which accepts exactly *one* vote (This can happen in

practice). The eccentricity value is 1, but the site is of no importance.

However, consider λ_{\max} itself. Obviously,

$$\frac{\lambda_{\min} + \lambda_{\max}}{2} \leq \lambda_{\max} \leq \lambda_{\min} + \lambda_{\max} \quad (7)$$

By (7) it is bounded from both sides by the proximity measure in (6) *and* has the eccentricity coded into it. When the value leans towards the left side of (7), eccentricity is low and vice-versa.

Thus, λ_{\max} is chosen as the raw saliency measure in our scheme.

This choice however, may still amplify locations which are very strong in terms of number of votes, but weak in eccentricity¹¹. The product of E and λ_{\max} produces the desired result, termed the *enhanced saliency* measure SM , or:

$$SM = \lambda_{\max} \cdot (1 - \lambda_{\min} / \lambda_{\max}) = \lambda_{\max} - \lambda_{\min} \quad (8)$$

Thus, $\lambda_{\max} - \lambda_{\min}$ is chosen as the enhanced saliency measure.

It is important to note that other functions of the eigenvalues can also satisfy the same conditions of monotonicity, but the ones chosen seem to be the *simplest* possible indicators of the desired behavior.

5 Detection of Junctions

A junction is defined as a *salient* point having a *low* eccentricity value.

Regular (non-junction) points along a curve are expected to have high eccentricity values. On the other hand, junction points are expected to have low eccentricity, since votes were accumulated from several different directions. By combining the eccentricity and the eigenvalue at a point, we acquire a continuous measure of the likelihood of that site being a junction. We redefine our previous definition of eccentricity slightly, so that low eccentricity scores high, or:

$$(E = \lambda_{\min} / \lambda_{\max}) \Rightarrow 0 \leq E \leq 1 \quad (9)$$

The product of our new eccentricity measure and the raw saliency measure λ_{\max} yields the junction saliency operator:

$$E \cdot \lambda_{\max} = (\lambda_{\min} / \lambda_{\max}) \cdot \lambda_{\max} = \lambda_{\min} \quad (10)$$

This process creates a *Junction Saliency map*. Interestingly enough, this map evaluates to just λ_{\min} at every site (as shown in (10)), which simply means that the *largest non-eccentric* sites are good candidates for junctions. By finding all *local maxima* of the junction map we localize junctions (see results in Figure 15).

10. Since $\lambda_{\min} \leq \lambda_{\max}$ and they are both non-negative.

11. For example, accumulation points and junctions! (where $\lambda_{\min} \approx \lambda_{\max}$)

6 Properties of the extension field

6.1 A longer line implies a stronger and more directed field, but up to a point.

Using a simple example we demonstrate the behavior of the field when extending a straight line. Figure 9 shows a cross-section of a saliency map computed on a series of straight lines with increasing lengths. Clearly,

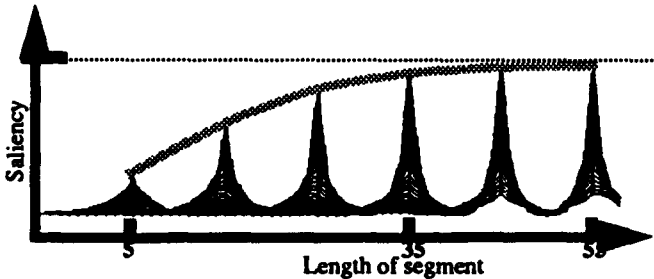


Figure 9 Saliency of a line as a function of length. It converges to some value which is the infinite straight integral of the extension field.

the saliency grows as a function of the length, and the map becomes more directed (thinner ridge). Also, saliency converges to some finite value which is just the infinite integral along the main axis of the Extension Field. This observation can be used to estimate absolute saliency.

6.2 The Non-maximum Suppression Phenomena

We have mentioned the superior selectivity of the Enhanced Saliency map. To illustrate this behavior, we look at the eccentricity *only* map of a straight line (Figure 10(a)) Note how low the eccentricity is close to

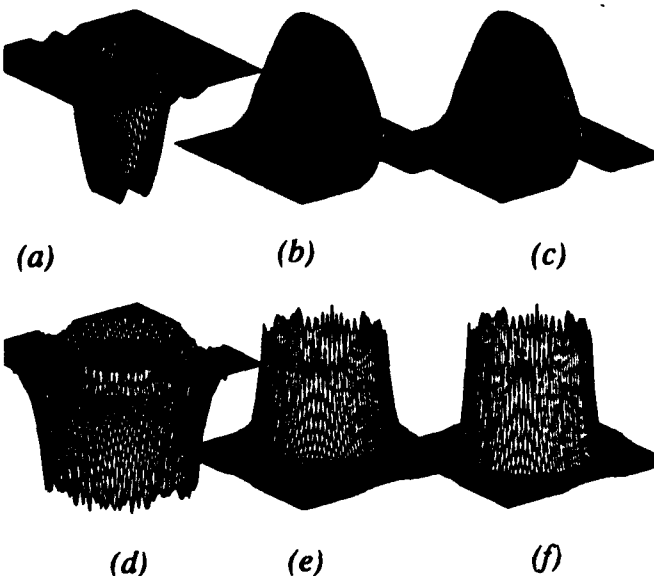


Figure 10 (a) Eccentricity only map of a saliency map of a straight line. (b) raw saliency map. (c) Product of (a) and (b). (d) - (f) same for a perfect circle.

where the real line passes. In Figure 10(b) we see the

raw saliency map of the same line. The Enhanced Saliency map is simply the product of the two maps, point by point, and will obviously sharpen the edges of the correct curves, thus creating a Non-maximum suppression affect (Figure 10(c)). Similar maps for a perfect circle are depicted in Figure 10(d-f).

The low eccentricity in the vicinity of the correct curve is due to the large variance of votes in these areas. These votes along the correct curve, when convolved, all vote for all sites, thus making the sites almost non-directional.

Contamination

It may look at first sight that the field, while voting for the correct curve, contaminates the environment by voting for many other cells in the image. This can be regarded as noise, and is inherent to the process. However, while the fields voting for a curve agree along the curve, they disagree in any other area of the image. This means that the contribution of a complete curve to the environment is almost *isotropic* and cannot affect the direction of true votes in any area. It has been shown in the previous section that close-by areas get low agreement values, and far away areas get low voting weight. In both cases, the interference is small.

This is true for all smooth curves in the image. Random segments¹² do contaminate the environment, and their effect is reduced through the robustness of the voting scheme. In the experimental results section we empirically test for allowable levels of noise.

7 Special Purpose Fields: The Straight Field

Special purpose fields are fields synthesized to enhance a special feature in an image. For example, in aerial images, a desired feature could be rectangular rooftops. It is possible to construct a saliency operator to enhance all straight line formations in the image, and at the same time suppress other smooth curves.

We now derive the shape of a field capable of finding all straight, or almost straight line formations in a cluttered directional edge image. This can be done in two ways.

The first method would just use the part of our original Extension Field that applies to straight lines. This simply means cutting off all field elements that are outside some pre-defined angle. The actual angle is of course a function of the amount of error one wishes to allow for a straight line.

A second and better motivated method would be to generate a new field (mask) by convolving with a piece of a straight line. This last method is more in the spirit of our previous special fields, and also retains the notions of feature fuzziness and continuum of saliency values. Again the length of the straight line with which

12. segments that should not belong to any curve

we convolve, will determine the amount of desired error in our definition of straightness.

8 Analysis of the End-point field

In many cases we tend to interpret end-points as being a partially occluded line. If enough end-points are available, they support the hypothesis of a shape occluding a collection of lines. Illusory edges appear to outline the said occluding shape. Figure 11 illustrates an end-point scenario.

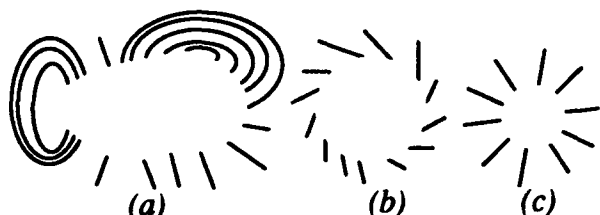


Figure 11 An end-point formation. (a) A center egg-like shape is not only perceived but also looks whiter (after [Boff, et al. 1986]). (b) An invisible circle occludes lines. No sensation of a circle is evident, because angles of intersection are not suitable. (c) The inner circle is perceived, but the outer one is not!

8.1 Straight angles in T-Junctions are more likely than any other

We derive the distribution of T-junction angles in a random world and show that close-to-straight T-junction angles are more likely to appear than any other angle. We claim that our human perception uses this property, and thus attempts to perform perceptual tasks on end-points stimuli only when the illusory intersection angles justify it. In Figure 11(a) all lines meet the illusory contour with almost straight angles, thus making the shape 'visible'. In Figure 11(b) the lines are occluded by an exact circle, but the angles are much more acute. No perception of shape is evident, even though the end-points trace an exact circle.

Claim: Given two unit size segments, we independently drop each one of them on a finite board, with uniform probability with respect to position and angle. We then look at the distribution of the intersection angles between the two segments. We claim that angles close to 90 degrees are more likely to appear than acute intersection angles. (We do not count cases where the segments do not intersect.) The proof can be found in [Guy and Medioni 1992b].

Curved objects can be approximated by linear segments to any degree of accuracy¹³ and should thus satisfy the same distribution. The same is true for lines of different lengths. It is easy to see that the length of the segment does not change the probability of a given intersection angle.

13. Non fractal curves.

8.2 Convex T-junctions are more common

We believe that the a priori probability of having convex T-junctions is larger than concave T-junctions. This observation can be illustrated through a simple example, as shown in Figure 12. Many other experiments

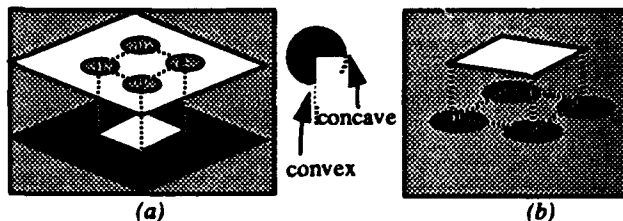


Figure 12 Convex and concave T-junctions. The Kanizsa square has two valid 3-D interpretations, but only one of them is perceived (b). The other one in (a) requires us to imagine concave T-junctions.

(such as in [Boff, et al. 1986]) indicate that end-point formations that require concave T-junctions are not normally perceived. Also, real objects tend to have longer convex boundaries, than concave ones, and since boundary length is proportional to the probability of an intersection, convex T-junctions are more common. We claim that humans use this property and tend to hypothesize mainly convex T-junctions.

8.3 Building the End-Point Field

The angle distribution derived previously suggests convolving our original Extension field with a multi-directional edge having a diameter function of a sinusoid, as was done for the point field, and shown in Figure 13.

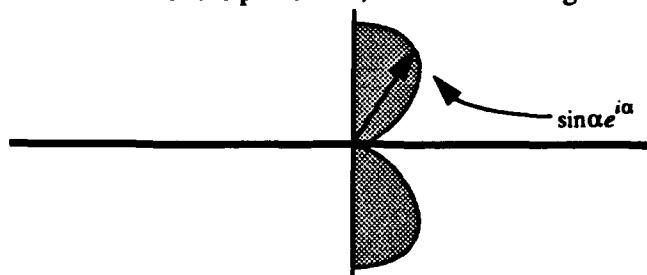


Figure 13 A multi-directional edge constructor for the End-Point Field. (envelope shown)

This kind of field will vote strongly for straight angle T-junctions, and weakly for other directions. This merely means that more support is needed for more acute angles. Results are given in section 10.

9 Complexity Issues

A naive way to implement the algorithm requires $O(n^2k)$ operations, where n is the side size of the image, and k is the number of edge elements in the input image. In practice, the local density of edgels restricts the useful scope of the field. This means that a smaller finite field can be used. The complexity becomes now $O(k)$. This last modification has the disadvantage of not being able to bridge gaps larger than the size of the field.

Alternatively, instead of computing a *dense* saliency map, we can compute the saliency of existing edgels only. This results in complexity of $O(k^2)$, and can be useful as a focus of attention map. This mode allows then for a second pass on the salient features only.

10 Results

10.1 On Synthetic Images

We have tested our approach with the synthetic data shown earlier in Figure 1. The saliency map produced is

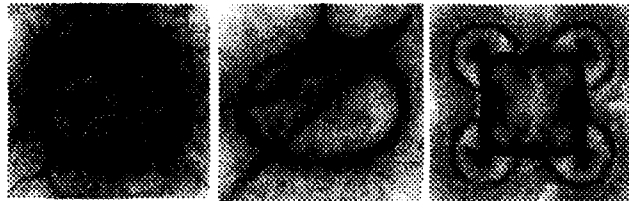


Figure 14 The Saliency maps of images in figure 1. shown (strength only) as grey-level image in figure 14 and the result of following the path of highest saliency produces a "clean" circle. Figure 14 also shows the result of the same procedure for the other scenes. Figure 15 shows an example of the steps involved in producing

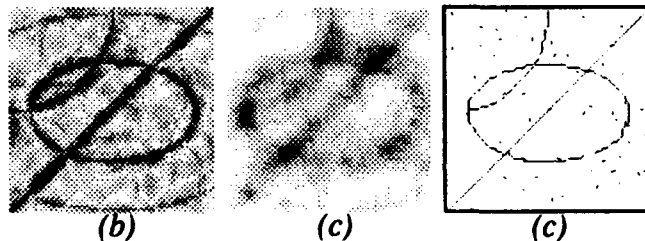


Figure 15 Extracting the most salient features. (a) Eccentricity enhanced map. (b) Junction saliency map, and (c) linking.

a high-level description of a given image, using the junction map in conjunction with the saliency map

10.2 Real Images

In figure 16 we show a real image example. The

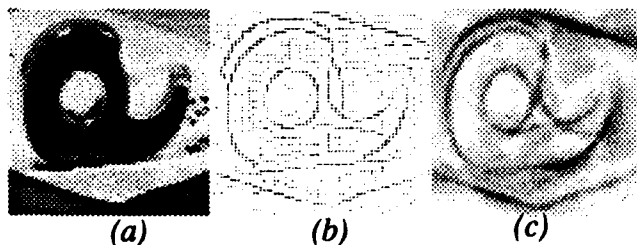


Figure 16 Example of a real image. (a) A tape dispenser image. (b) All edges. (c) Eccentricity enhanced saliency map.

original image was processed with a simple edge detector (5x5 step masks), without any linking. Note that the edge image is fragmented and has a lot of noisy segments. Figure 16(c) shows the resulting Saliency map.

10.3 Point Field

We tested our system on the image in figure 17(a). Initially, the system was run using the Point field. This resulted in a saliency map *with* orientation data. A second phase of computation was then performed, using the directional Extension field (Figure 1). That stage produced the final saliency map as shown in figure 17(b).

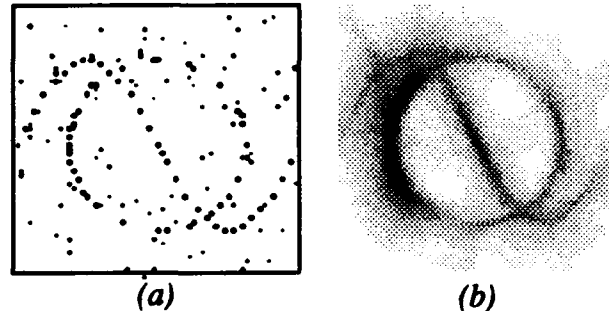


Figure 17 (a) A non-directional input image. (b) Saliency map, after applying the Point field and the directional extension field

10.4 Straight Field

We tested our straight line operator on the previous example, as shown in Figure 17. Clearly the straight

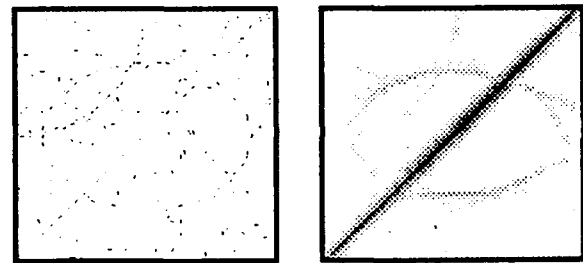


Figure 18 The enhanced saliency map of the straight field

line appears to be salient. The ellipse is still vaguely visible, since it can be viewed as being straight to some degree.

10.5 Noise Breakdown Point

Since we want our algorithm to mimic human perceptual capabilities, we actually prefer to fail where humans fail, unlike the Hough transform, for example, which may find line formations even in a very cluttered image, where such formations are not visible, and could be a result of accidental alignments.

We have performed a controlled experiment to determine the amount of random noise which still allows for a correct following of a curve, given a constant factor of existing edge. The test is whether all points along the perceptual circle belong to the local maxima, as defined before. Only uniform noise is space and orientation was applied to the original image. The circle has a radius of 25 pixels and consists of 33 segments and the percentages of noise applied are 2, 4, and 6 percent. This

means that about a quarter of the circle edge exists. Up to 4 percent of noise, the circle (or large parts of it) is recoverable. With 5 percent noise and more, the saliency map degrades, and it is no longer possible to apply the following algorithm. More detailed results can be found in [Guy and Medioni 1992b].

10.6 The End-Point field

We tested the end-point field with the synthetic image in Figure 11. It is clear that the outer circle receives

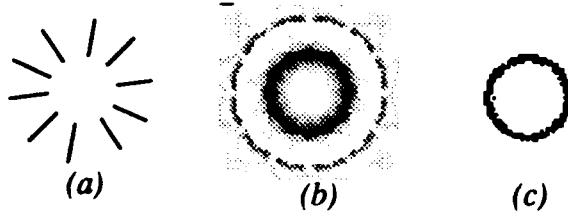


Figure 19 Results of applying the end-point field. Note that the outer circle is not highlighted! (a) Original image. (b) Saliency map. (c) after thresholding single votes.

relatively low saliency, and can be completely removed if we consider single votes in the plane as wrong hypotheses (see Figure 20).

11 Comparison with the Hough Transform

The classical Hough transform ([Duda and Hart 1972],[Hough 1962]) was invented to detect co-linear formations within a *dot image* in the presence of noise.

11.1 The classical Hough transform

Consider the Hough transform for straight lines. A 2D array of accumulators is constructed and each token in the image votes for a family of straight lines which may pass through that dot. Using the common (θ, d) line parameterization it is easy to show that the voting pattern is a *sinusoid*.

The next phase in the process is to search for peaks in the parameter space array. That line is only defined in terms of the above two parameters and further processing is needed to localize the actual segment in the image plane. Note that this process is absolutely global, as it ignores distances between contributing candidates. If some orientation data is known at a site, less candidates are voted for, and the ability to reveal the desired shape gets better.

The same scheme can be extended for other shapes [Ballard 1981], by extending the parameter space.

11.2 Our scheme as a Hough transform

Our system can be viewed as a Hough Transform where *the parameter space is the image plane itself*. This allows for many more degrees of freedom in the choice of shapes, and in the basic definition of the desired shapes. This choice of the parameter space allows us to define other voting patterns¹⁴ which enable us to encode the constraints (see 2.3.1).

Our scheme is thus capable of finding shapes described by their properties (smoothness etc.) rather than by their exact analytical parameters (as in the Hough Transform).

For example, to implement a circle finder using our scheme, we would simply use our Extension Field, but with all field strengths set to the value 1. Coding of the other constraints is where the strengths of the individual field elements come into play. *This is not possible with the original Hough Transform.*

The down side is that the result is not found at an isolated peak of the parameter space, but rather as a continuous ridge of peaks (in the case of the Extension Field). Also, note that the parameter field we use is *always* the image plane, which, in many cases, is smaller than a classical Hough parameter space. This 'Compaction' of data is the cause for the low selectivity compared to Hough transforms.

Table 2: Comparison between the Hough Transform and Our Scheme

	Our scheme	Hough Transform
Complexity	Same for all shapes and properties	Grows with the number of parameters
Space Requirements	Constant ($O(\text{image size})$)	Grows with the number of parameters and resolution.
Properties (Families of shapes) coding	Yes, by defining suitable fields	Not in any obvious way
Analytical shapes	Yes, by the same method	Yes
Voting pattern ^a	2-D	1-D ^b
Localization of shape	Yes	Not always. Further processing is needed to find location
Means of output	A ridge of maxima	A peak in parameter space
Choice of bin size	Always image resolution	Hard (May be crucial)
Selectivity	Low (Because of compaction)	High
Transformation	Non-Linear	Linear

a. Our nomenclature. Refer to text for definition.

b. Or at least one dimension less than that of the problem.

14. Similar to the sinusoid defined for the straight line detector

However, for Perceptual Grouping this loss of selectivity seems to be an advantage. Note that the Hough Transform could find formations (e.g. straight lines) even when they are *not* salient to humans, because the notion of interfering features does not exist¹⁵. This makes the Hough Transform a *linear Transformation* which is not suitable for perceptual grouping. Our scheme, on the other hand, is highly non-linear and takes into account the interference by computing the eccentricity measure at each site. Table 2 summarizes the main differences and similarities of the two methods.

12 Summary And Conclusion

We have introduced a unified way to extract perceptual features in edge images. By 'unified' we mean that all low-level features (edgels, points) are treated in a uniform way, and no special cases exist. The scheme is threshold-free and non-iterative. It is especially suitable for parallel implementation, since computations of the saliency maps are independent for each site, and parallel algorithms for line following are known and can easily be adapted. Also, calculations are simple and stable, as no curvatures or any other derivatives need to be computed on the digital curves.

The system can rank features based on their perceptual importance. This allows a real-time application to process as many features as time permits.

Some of the issues which have not been addressed are the resolution dependency of the description. At this time, only one level of description is possible. Also, we have not tried to localize end-points of curves ending abruptly. Since all computations are performed on a discrete grid, quantization and rounding errors restrict the selectivity and amount of clutter the system can handle.

References

- [Ahuja and Tuceryan 1989]N. Ahuja and M. Tuceryan, *Extraction of early perceptual structure in dot patterns: integrating region, boundary, and component Gestalt*, CVGIP 48, 1989, pp. 304-356.
- [Ballard 1981]D. H. Ballard, *Generalizing the Hough Transform to Detect Arbitrary Shapes*, Pattern Recognition 13, 2, 1981, pp. 111-122.
- [Boff, et al. 1986]K. R. Boff, L. Kaufman, and J. P. Thomas, *Handbook of Perception and Human performance*, Vol. II, John Wiley and Sons, 1986, pp. 36-1 - 36-30.
- [Clowes 1971]M. B. Clowes, *On Seeing Things*, Artificial Intelligence 2, 1, 1971, pp. 76-116.
- [Dolan and Weiss 1989]J. Dolan and R. Weiss, *Perceptual Grouping of Curved Lines*, Proc. IUW89, Palo Alto, CA., pp. 1135-1145.
- [Duda and Hart 1972]R. O. Duda and P. E. Hart, *Use of the Hough Transformation to Detect Lines and Curves in Pictures*, Communications of the ACM, Vol 15, 1972, pp. 11-15.
- [Guy and Medioni 1992a]G. Guy and G. Medioni, *Perceptual Grouping using Global Saliency enhancing operators*, Proc. of ICPR92, The Hague, Holland, 1992, pp. 99-104
- [Guy and Medioni 1992b]G. Guy and G. Medioni, *Perceptual Grouping using Global Saliency enhancing operators*, IRIS-USC Technical report, to appear.
- [Helson 1933]H. Helson, *The Fundamental Propositions of Gestalt Psychology*, Psychological Review, 1933, 40, pp. 13-32.
- [Hough 1962]P.V.C. Hough, *A Method and Means for Recognizing Complex Patterns*, U.S. Patent No. 3,069,654, 1962.
- [Kanizsa 1976]G. K. Kanizsa, *Subjective contours*, Scientific American, April 1976.
- [Lowe 1987]D.G. Lowe, *Three-dimensional object recognition from single two-dimensional images*, Artificial Intelligence 31, 1987, 355-395.
- [Mohan and Nevatia 1989a]R. Mohan and R. Nevatia, *Segmentation and description based on perceptual organization*, Proc. CVPR, Jun. 1989, San Diego, Ca., pp. 333-341.
- [Mohan and Nevatia 1989b]R. Mohan and R. Nevatia, *Using Perceptual Organization to Extract 3-D Structures*, IEEE Trans. on PAMI, Vol. 11, No. 11, November 1989, pp. 1121-1139.
- [Parvin and Medioni 1991]B. Parvin and G. Medioni, *A Dynamic System for Object Description and Correspondence*, Proc. CVPR, Jun. 1991, Maui, Hawaii, pp. 393-399.
- [Rom and Medioni 1993]H. Rom and G. Medioni, *Hierarchical Decomposition and Axial Shape Description*, IEEE Trans. on PAMI, to appear.
- [Sha'ashua and Ullman 1988]A. Sha'ashua and S. Ullman, *Structural saliency: the detection of globally salient structures using a locally connected network*, Proc. ICCV, Dec. 1988, Tampa, FL., pp. 321-327.
- [Stein and Medioni 1992]F. Stein and G. Medioni, *Recognizing 3D Objects from 2D Groupings*, IUW92, Jan. 1992, San Diego, Ca., pp. 667-674.
- [Sugihara 1984]K. Sugihara, *An Algebraic Approach to Shape-From-Image Problems*, Artificial Intelligence, Vol. 23, No. 1, 1984, pp. 59-95.
- [Ulupinar 1991]F. Ulupinar, *Perception of 3-D Shape from 2-D Image of Contours*, Ph.D. Thesis, USC, August 1991.
- [Waltz 1975]D. I. Waltz, *Generating Semantic Descriptions from Drawings of Scenes with Shadows*, Chapter 3 in *The Psychology of Computer Vision*, P. H. Winston (Ed.), Mc-Graw Hill, New York, 1975.
- [Zucker et al. 1988]S. W. Zucker, A. Dobbins, C. David, and L. Iverson, *The Organization of Curve Detection: coarse Tangen: Fields and Fine spline Coverings*, Proc. ICCV, Dec. 1988, Tampa, FL., pp. 568-577.

15. Under the assumption of random background. Some repetitive patterns may interfere.

A TRANSFORM FOR DETECTION OF MULTISCALE IMAGE STRUCTURE

Narendra Ahuja

Beckman Institute and Coordinated Science Laboratory

University of Illinois

405 North Mathews Avenue

Urbana, Illinois 61801

Abstract

This paper describes a new transform to extract the edge contours and skeletons of image regions at multiple scales. The application of the transform to detecting edge structure is explained in detail. It is argued that linear processing based approaches, such as convolution and matching, have the fundamental deficiency of using a priori models of edge geometry. The proposed transform avoids this limitation by letting the structure "emerge," bottom-up, from interactions among pixels, in analogy with statistical mechanics and particle physics. The transform involves global computations on pairs of pixels followed by vector integration of the results, rather than scalar local linear processing. An attraction force field is computed over the image. Pixels belonging to the same region are mutually attracted whereas those across edges repel each other. Scale is an integral parameter of the force computation. The resulting groupings of pixels represent multiscale image structure. The properties desired in multiscale edge detection are given, and it is theoretically and experimentally shown that the transform possesses these properties. Along with their contours, the transform also extracts skeletons of multiscale regions. Experimental results with synthetic and real images are given to demonstrate the properties of the transform.

1 Introduction

This paper is concerned with the problem of detecting low level structure in images. It introduces a transform to facilitate integrated edge and region detection at all geometric and photometric scales at which structure is present in a given image. At each scale, the transform helps identify groupings of pixels which comprise homogeneous regions surrounded by edges by distinguishing

the locations of region edges and skeletons. It is argued that the usual convolution and matching approach has inherent limitations in edge detection. The main motivation and contribution of the approach is to avoid a priori models of edge geometry but still allow detection of region boundaries with arbitrary curvature. Instead of searching for and fitting spatial models of edges and regions over image neighborhoods to select edge locations and orientations, the structure is allowed to "emerge" from "interactions" among the pixels. A pixel's interaction with all other pixels is considered instead of testing specific neighborhoods for a specific structure.

A central feature of the proposed transform is to compute affinities of pixels for grouping with other pixels, such that the groupings reflect the image structure. For pixels on either side of a region boundary, the transformation yields high affinities, but there is little affinity between pixels across regions. The strength of interaction, and therefore affinities, between pixels depend upon their distances and contrasts, and this relates the computed affinities to the different scales present. Due to the global interpixel interaction allowed, the transform can be viewed as collecting spatially distributed evidence for edges and skeletons and making it available at their locations. In this sense, the transform performs Gestalt analysis. The transform brings out the image structure which makes its subsequent detection easier. The objective of this paper is to introduce the transform; algorithms for structure detection from the transformed image are beyond its scope.

In the rest of this paper, we will first illustrate in detail the use of the transform for edge detection, and then briefly discuss its use for region skeleton extraction. For concreteness, we will assume that image regions have uniform gray levels and are surrounded by step edges, although the approach extends to other types of regions and edges. Section 2 discusses some basic desired characteristics of edge detection and the motivation for the proposed approach. Section 3 proposes a family of transforms intended to achieve the above characteristics, and describes some of its properties of interest. Section 4 proposes a specific transform from the family and examines its performance for edge detection in

*This work was supported by the Defense Advanced Research Projects Agency and the National Science Foundation under grant IRI-8902728. Thanks to Mark Tabb who produced the experimental results included in this paper.

greater detail. Section 5 briefly discusses the application of the transform to extracting skeletons of regions. Section 6 describes some experiments conducted to test the performance of the transform.

2 Background and Motivation

This section explains the motivation behind the approach presented in this paper. For concreteness, we focus on the problem of edge detection. The arguments extend to the dual case of region detection which is discussed in Sec. 6. We first examine the critical aspects of the performance of an edge detector (Sec. 2.1) which suggests certain basic desired characteristics of multiscale edge detection (Sec. 2.2). Then we explain salient design features of the approach proposed in this paper, and how they help meet the desired performance characteristics.

2.1 Two Aspects of Edge Detection

There are two major aspects of edge detection which are of central importance to its performance. The first one has to do with the photometric and geometric model of an edge. An edge separates two different regions and thus two different types of gray level populations. It is common to treat the problem of edge detection as mainly that of selecting a point along the intensity profile across edge, assuming such a profile can be extracted from the image. Accordingly, a model of the intensity profile is used to precisely define an edge and to optimally detect its location. Different types of intensity models of an edge have been proposed, according to the nature of the two populations and the spatial profile of the transition from one to the other across the edge [13, 12, 3, 5]. To meet the assumption that edge profile through a pixel can be identified, it is common in edge detection work to use a model of edge curvature. The requirement for the identification of edge profile may be explicit or implicit. The geometric model constrains the number of possible ways in which subdivisions of the pixel neighborhood into two regions can be made. Each subdivision must be implicitly or explicitly tested for the presence of an edge profile in some direction. For example, the assumption of local straightness of edge is common which makes it very easy to select neighborhoods on the two sides of an edge. [10] assumes that the edge is locally straight (and that the intensity changes linearly along a direction parallel to the edge.) Nalwa and Binford [11] assume straightness to extract a sample edge profile. Even the computation of gradient which is common to many edge detectors [9] implicitly assumes local edge straightness. The same can be said about Laplacian based edge detectors. The use of straightness is very explicit in the different types of discrete edge masks each of which is meant to detect a different edge orientation [14]. To detect intensity facets meeting at an edge [6], a model of edge geometry is required so candidate neighborhoods from each side of the edge can be identified. The work on optimal edge detection (e.g.,

[4]) is thus also subject to the validity of the assumed model of the edge geometry.

The issue of the validity of the models of edge profile has been addressed and different models of edge (step, ramp, roof) have been mentioned. However, the limitations and impact of the assumptions made about edge geometry have received much less attention. These assumptions cause significant errors in the results as can be seen, for example, from the performance of the Laplacian-of-Gaussian for different edge geometries [2].

The second major aspect of edge detection is related to scale [15, 16, 8]. There are two important ways in which edges are associated with scale: geometric and photometric. One example of geometric scale is structure size. Only large structures may be visible at a coarse geometric scale while smaller sizes and features may be detected at finer scales. An analogous process characterizes contrast across an edge. An edge contour which separates two regions of a given contrast at a given scale may not be detected at a higher scale. Thus, there are scale variations associated with both geometric and photometric sensitivity to detail. The exact number and parameters of scales present in a given image is a priori unknown. Therefore, for an edge detector to work at multiple scales, it should automatically identify the scales present and detect edges corresponding to each scale.

2.2 Desired Characteristics of Multiscale Edge Detection

The above discussion leads us to the following desired characteristics of an edge detector.

A. Shape Invariance: The edge should be correctly detected regardless of the local curvature. Thus, an edge point must be detected at only one and correct location, regardless of whether the edge in the vicinity of the point is straight, curved or even contains a corner.

B. Contrast Scaling: It should be possible to detect edges according to their contrast. For example, as scale increases, the required contrast of detectable edges may increase.

C. Geometric Scaling: It should be possible to detect edges of regions according to their sizes. For example, as scale increases the required size of detectable geometric features may increase.

D. Stability and Automatic Scale Selection: Image structures at different scales correspond to locally invariant (stable) descriptions with respect to geometric and contrast sensitivities. Since for an arbitrary image these scales are a priori unknown, they should be identified automatically.

2.3 Limitations of Convolution, and the Proposed Approach

The approach we present in this paper has been motivated by the desire to satisfactorily address both aspects of edge detection discussed in Sec. 2.1 and to achieve the desired characteristics listed in Sec 2.2.

The basic limitations put on the performance of an edge detector by the use of models of edge geometry suggest that no linear, convolution based approach could be satisfactory. This is because the convolution kernel represents a template for the edge, expected edge geometry. In the digital case, one could attempt to circumvent this problem by exhaustively considering all possible edge geometries in a neighborhood. But the number of resulting kernels will fast increase with neighborhood size and will be prohibitively large for any reasonable size neighborhood.

The approach presented in this paper uses a method involving computations on pairs of pixels followed by vector integration of the results, rather than scalar, weighted averaging over pixel neighborhoods. This avoids making assumptions about edge geometry. The inspiration for the proposed solution comes from physics where microscopic homogeneity of physical properties leads to islands of, say, similar particles or molecules. An island shape is congruent with the space occupied by a set of contiguous, similar particles, whatever the complexity of the boundary! The particles group together and coalesce into regions based on the similarity of their physical property only. The homogeneity of the physical property then characterizes the resulting regions. As an alternate analogy, the grouping process is like the alignment of microscopic domains in large areas of a ferromagnetic material, with a reversal of magnetization direction taking place across the boundary between two similar poles facing each other. The key process is that of interaction among particles, which leads to bindings among them based on their physical properties.

The problem of edge detection has similarities to the above physical process. The goal is to find a partition of the image, regardless of the boundary complexity, such that each cell of the partition is homogeneous, say, in gray level. The physical analogy suggests a formulation of the edge detection process in terms of a suitably defined process of interpoint interaction - a process that would group, bottom-up, each set of points of the same property, say gray level, corresponding to a region. Since points are the primitives of structure, they could group together to follow any region boundary. Being a parameter of the grouping process, different degrees of acceptable homogeneity within a region would yield groupings over different regions, making scale an integral part of structure detection.

In the next section, we propose a family of transforms which achieves the above grouping. These transforms can be viewed as yielding an attraction-force field in which all points are mutually attracted except those across edges.

3 A Family of Transforms for Multiscale Edge Detection

In this section we first describe a family of transforms and discuss how these transforms possess the desired characteristics listed in Section 2. The discussion

throughout this section will address general properties, in qualitative terms, since the family of transforms has only been characterized in general terms. A specific transform from this family will be examined in Section 4 with more quantitative performance analysis. To simplify the presentation, the regions are defined as being homogeneous in gray level. We will consider the case of uniform gray level regions whenever doing so simplifies analysis without loss of applicability of the results to the case of regions having only statistical homogeneity.

3.1 A Family of Transforms

Consider a transform over the image which computes an attraction-force field wherein the force at each point denotes its net affinity to the rest of the image. The force vector points in the direction in which the point experiences a net attraction from the points in the rest of the image. For example, a point inside a region would experience a force towards the interior of the region. This force is computed as the resultant of attraction-forces due to all other image points. Let $F(p, q)$ denote the magnitude of the force vector $\mathbf{F}(p, q)$ with which a pixel P at position p is attracted by another pixel Q at location q . Thus, $\mathbf{F}(p, q) = F(p, q)\hat{\mathbf{r}}_{pq}$, where $\hat{\mathbf{r}}_{pq}$ denotes the unit vector in the direction from P to Q , i.e.,

$$\hat{\mathbf{r}}_{pq} = \frac{\mathbf{q} - \mathbf{p}}{\|\mathbf{q} - \mathbf{p}\|}$$

In the continuous image plane, an image is transformed into a continuous vector field. The resultant attractive force vector \mathbf{F}_p at P is given by

$$\mathbf{F}_p = \int_{q \in \text{Image}} F(p, q)\hat{\mathbf{r}}_{pq} dq \quad (1)$$

In the discrete case,

$$\mathbf{F}_p = \sum_{q \in \text{Image}} F(p, q)\hat{\mathbf{r}}_{pq}$$

We must now specify what forms could the force function $F(p, q)$ take. We will do so by considering the properties that \mathbf{F} must possess. The properties will define a family of transforms. Any choice of \mathbf{F} that has these necessary properties will suffice. A specific choice of \mathbf{F} will yield one member of this family.

Since the presence of an edge must be determined by its immediate vicinity (adjoining regions) rather than by distant points across other intervening regions, the force should be a decreasing function of distance. This will be accomplished by making the force exerted on a given pixel P by another pixel Q to be inversely proportional to the distance between P and Q . Further, a pixel should be attracted more to a pixel within its own region than to one in a different region. This is accomplished by making the force to be inversely proportional to the difference between the gray levels of P and Q . The rate at which the force decreases with distance determines

the geometric scale (size) of the regions whose structure is reflected in the force field. Similarly, the rate at which the force decreases with gray level difference determines the photometric scale captured by the force field. We will use σ_s and σ_g as the geometric and gray level scale parameters. The larger their values, the larger will be the force on a pixel due to another pixel having a given distance and gray level difference, respectively.

We will now examine some basic properties of the force field \mathbf{F} . These properties serve to illustrate qualitatively the basic motivation behind the use of the family of transforms of Eqn (1). More quantitative and detailed evaluation of the performance will be presented for a specific transform in Section 3.

1. Magnitude and Directionality: The magnitude of \mathbf{F}_p increases as P moves closer to the region boundary. The direction of \mathbf{F}_p points toward region interior.

Proof:

Consider a homogeneous gray level region W surrounded by another region X of gray level contrast C . Now consider an arbitrary point P inside W and a disk of radius r centered at P . For sufficiently small values of r , the disk is contained inside W , and because of gray level isotropy around P , the resultant force on P due to the pixels inside the disk is 0. Figure 1a illustrates this for the simple case where W is a half plane. As r increases to some value R , the disk will touch W . For $r > R$, the force on P due to the pixels inside the disk will be nonzero because of asymmetry. Due to the gray level difference between P and the pixels outside W , there will be a net reduction in the force on P from the direction of intersection with X , and therefore, the net force on P will point away from the region of intersection (Fig. 1a). That is, the resultant force \mathbf{F}_p on P and the direction PQ will satisfy the condition $\mathbf{F}_p \cdot \hat{\mathbf{r}}_{pq} < 0$. For a region boundary having a simple shape around the point nearest to P , the larger $(r-R)$ is, the larger will be the anisotropy, and hence, the larger the magnitude of \mathbf{F}_p . Therefore, for any fixed $r > R$, the closer the point P is to the border (i.e., the smaller R is), the larger the magnitude of \mathbf{F}_p . As P moves closer to the boundary the force continues to point away from X and its magnitude increases.

For regions having more complicated shapes than a rectangle, as r increases there will in general be multiple connected components of intersection between the disk and the surrounding regions, due to the complex shape of the region boundary. Therefore, the rate of decrease in the value of \mathbf{F}_p will depend on the exact shape of W 's boundary. Fig. 1 illustrates \mathbf{F}_p for the case of a rectangular region (Fig. 1b) and an arbitrary region (Fig. 1c).

2. Orthogonality: Consider a point P just inside the boundary of a region W , such that the boundary within

a disk D of radius r centered at P , $r \gg \sigma_s$, is symmetric about P . Then the direction of \mathbf{F}_p points into the region and is normal to the boundary at P .

Proof:

Since $r \gg \sigma_s$ by assumption, the force \mathbf{F}_p at P gets little influence from the points outside disk D . Consider a point Q inside the region as well as D , and along the normal to the boundary at P . Since the boundary of W within D is symmetric about P (Fig. 2), the line through P and Q divides the region of intersection of W and D into two symmetric halves. For each point M in one half, there exists another point M' in the other half such that the components of the force at Q , orthogonal to PQ , due to M and M' are equal and opposite. Thus, the net force at all points along PQ , including P , is along PQ and, from the directionality property, away from the boundary.

3. Smoothness: If $F(p, q)$ is a continuous (or differentiable) function for any p and q , and the intensity value within any image region is continuous (or differentiable), then so is \mathbf{F} at all nonboundary points of the region.

Proof:

Consider a nonboundary point. Then \mathbf{F}_p is given by

$$\int F(p, q) \hat{\mathbf{r}}_{pq} dq$$

. Consider a point at $p + \delta p$ in the vicinity of p . Then the differential change $\delta \mathbf{F}$ from location p to location $p + \delta p$ is determined by the corresponding differential changes $\delta F(p, q)$ and $\delta \hat{\mathbf{r}}_{pq}$. Now, $\delta F(p, q)$ is given, from its definition, in terms of the change $\delta d(p, q)$ in distance to q , and the change $\delta g(p, q)$ in gray level relative to q . $\delta \hat{\mathbf{r}}_{pq}$ simply depends on p . Since $g(p, q)$ is given to be continuous (differentiable) away from the region boundary, and $d(p, q)$ and $\hat{\mathbf{r}}_{pq}$ are certainly continuous (differentiable) functions of p , \mathbf{F} is also continuous (differentiable) with respect to p away from the region boundary. Across a region boundary, the gray level variation is discontinuous, and therefore, so is \mathbf{F} .

3.2 Multiscale Edge Detection

In this section, we show that the field defined by \mathbf{F} makes the interregion edges explicit and this makes the task of edge detection easier. Consider an edge point P on the border of a region W with another region X , and a neighborhood around P of radius r . Further assume that the conditions of Property 2 are met, namely, $r \gg \sigma_s$, and W 's boundary is symmetric about P within the neighborhood. Then, from Property 2 above, as P is approached from the side of W , the direction of \mathbf{F} is orthogonal to the edge at P and points toward W . Similarly, from Property 2, as P is approached from the side of W' , the direction of \mathbf{F} is orthogonal to the edge and points toward W' . Thus P is a point of directional discontinuity, and the direction changes by $\pi/2$

while traversing from W into X. Further, recall that from Property 3, \mathbf{F} varies smoothly on either side of the edge. Therefore, for each point of directional discontinuity, there is another point P' in the vicinity of P which is also a point of directional discontinuity. Since with appropriate choice of scale parameter each such point is along the region border (true edge), and regions have closed borders, the points of directional discontinuity found using appropriate scale parameter form closed curves.

These properties also hold near vertices where more than two regions meet. This is true because the force at each point in the vicinity of a vertex points toward the interior of one of the regions, and the above arguments hold for each region. Further, any variation in the values of σ_s results in incremental changes in \mathbf{F} due to the smoothness property. These observations lead us to the following result:

RESULT: Region borders are characterized by a discontinuity in the direction of \mathbf{F} . The magnitude of the discontinuity is $\pi/2$ for optimal choice of scale parameters at each point and decreases gradually for suboptimal choices.

3.3 Performance Analysis

We will now examine the performance of \mathbf{F} with respect to the desired characteristics listed in Section 2.2. First, the very motivation for proposing \mathbf{F} comes from characteristic A, namely, invariance to local edge geometry. Sec. 3.2 explains how this characteristic is possessed by \mathbf{F} . With regard to desired characteristic B, scale parameter σ_g provides a mechanism to accomplish contrast scaling. As σ_g increases, adjacent regions may merge. This is because the attraction of a point in one region from another point across region boundary may increase sufficiently so that the directional discontinuity in \mathbf{F} responsible for the edge may vanish. Thus changing σ_g achieves region split and merge based on contrast values [7], and therefore, desired contrast scaling. Analogously, scale parameter σ_s helps achieve geometric scaling. As σ_s increases, the force field at a point starts to get influenced by increasingly global structure. Therefore, as σ_s increases, the sensitivity to fine local details is reduced resulting in a characterization of more global shape.

We now consider the performance with respect to desired characteristic D. First, recall that the region edges are represented by discontinuities of \mathbf{F} . From the above discussion, the structures at different scales must be detected by appropriate values of (σ_s, σ_g) . Since there could be only a small number of scales associated with any specific area of the image, the parts of the (σ_s, σ_g) space associated with the different scale structures in any given area must be small in number. For any arbitrary σ_s and σ_g , the force field should have a direction pattern in which borders of only certain regions coincide with direction discontinuities. This force field direction pattern should not change except when a change

in σ_s leads to the emergence/loss of a structure due to changed geometric sensitivity, or a change in σ_g leads to a split/merge of regions due to changed contrast sensitivity. Consequently, changes over much of the range of σ_s and σ_g should preserve the force direction pattern in any given image area. This implies that \mathbf{F} should be stable and the significant scales in any image area could be automatically identified as corresponding to those σ_s, σ_g values (unhashed areas in Fig. 3) for which there is little change in the direction pattern. Such scale identification should be quite robust since it is based upon qualitative changes in \mathbf{F} .

4 A Proposed Transform

In this section we describe a specific transform belonging to the family defined by Eqn (1), and examine its performance as a multiscale edge detector.

4.1 Transform

We define $F(\mathbf{p}, \mathbf{q})$ in Eqn (1) as a product of two Gaussians, one a decreasing function of the distance $d(\mathbf{p}, \mathbf{q})$ between P and Q , and the other a decreasing function of the gray level difference, $g(\mathbf{p}, \mathbf{q})$ between P and Q . The standard deviation for the spatial variation is the spatial scale parameter σ_s , and the standard deviation for the gray level variation is, the scale parameter for gray level difference, σ_g . The choice of Gaussian for each part is made mainly because of its optimal localization properties in both spatial and transform domains, although other properties such as separability are also desirable.

Therefore, the proposed transform \mathbf{Fp} at an image point P at location \mathbf{p} is defined by:

$$\mathbf{Fp} = \int_{\mathbf{q} \in \text{Image}} e^{-\frac{d^2(\mathbf{p}, \mathbf{q})}{2\sigma_s^2}} e^{-\frac{\Delta g^2(\mathbf{p}, \mathbf{q})}{2\sigma_g^2}} \hat{\mathbf{r}}_{\mathbf{p}\mathbf{q}} d\mathbf{q} \quad (2)$$

As we will see later, empirically the performance of the transform for edge detection seems to be quite insensitive to the nature of the decreasing function; the formulation of interpixel interaction as a vector integration of pairwise pixel similarities, rather than scalar weighted averaging over neighborhoods, appears to be the key factor in capturing the image structure.

4.2 Performance Analysis

In this section, we will examine the performance of the transform of Eqn. (2) for edge detection with respect to the various desirable characteristics listed in Section 2. Specifically, we will discuss the performance of the proposed transform with respect to (A) sensitivity to edge geometry, (B) sensitivity to intensity model, and (C) multiscale behavior.

A. Sensitivity to Edge Geometry: For an edge through a point P which is smooth in the vicinity of P , we have seen that the edge is well represented as a directional discontinuity in \mathbf{F} . That is, if the edge is

smooth around P and σ_s is small relative to the radius of curvature of the edge contour, the transform captures the edge information. However, if the curvature assumes a high value or is undefined near P, e.g., when a corner is present along the edge close to P, the behavior of the transform needs to be examined. This is because there may be a qualitative change in F_p as σ_s increases and begins to include significant contributions from the parts of edge beyond the corner (or high curvature) part of the edge. Consider the corner of angle θ shown in Fig. 4. Point P is located a distance d away from the corner. As σ_s increases, F changes significantly, from that corresponding to a long linear edge, to that corresponding to the boundary of a wedge shaped region.

To study the performance of the transform on a corner, we attempted to derive a closed form expression for F as a function d, θ and σ_s and σ_g . Due to the use of the Gaussian, some of the integrals did not appear to have closed form solutions. However, we could derive a closed form expression for the case of a linear decay (instead of Gaussian) in attractive force with distance. It is shown in [1] that for the case $R \leq \sigma_s$ and $d < R$ (Fig. 4), the components F_x and F_y of F are given by

$$-F_x = (F_{oL} + F'_{oL})(1 - e^{-\frac{c^2}{2\sigma_s^2}})$$

$$F_y = (F_{\frac{\pi}{2}L} + F'_{\frac{\pi}{2}L}) \left(1 - e^{-\frac{c^2}{2\sigma_s^2}}\right)$$

where $F_{oL} + F'_{oL}$

$$= R \left(1 - \frac{R}{2\sigma_s}\right) \sin \theta \left[\sqrt{1 - (d^2 \sin^2 \theta)/R^2} + \frac{d \cos \theta}{R} \right] + d \sin \theta \left[\cos \theta \ln \left(\frac{R}{d}\right) - \frac{d \sin^2 \theta}{2\sigma_s} \ln \left(\frac{d(1 - \cos \theta)}{R(1 + \sqrt{1 - \frac{d^2 \sin^2 \theta}{R^2}})} \right) \right]$$

$$+ \frac{\cos \theta}{2\sigma_s} \{d - R\} + \sin \theta \left\{ \sin^{-1} \left(\frac{d \sin \theta}{R} \right) - \pi + \theta \right\}$$

and $F_{\frac{\pi}{2}L} + F'_{\frac{\pi}{2}L}$

$$= R \left[1 - \frac{R}{2\sigma_s}\right] \left[1 - \cos \theta \sqrt{1 - \frac{d^2 \sin^2 \theta}{R^2}} + \frac{d \sin^2 \theta}{R} \right]$$

$$+ d \sin \theta \left[\sin \theta \ln \left(\frac{R}{d}\right) + \right.$$

$$\left. \frac{d \sin \theta \cos \theta}{2\sigma_s} \ln \left(\frac{d(1 - \cos \theta)}{R(1 + \sqrt{1 - \frac{d^2 \sin^2 \theta}{R^2}})} \right) \right]$$

$$+ \frac{\sin \theta}{2\sigma_s} (d - R) - \cos \theta \left\{ \sin^{-1} \left(\frac{d \sin \theta}{R} \right) - \pi + \theta \right\}$$

These components determine the magnitude and the direction of the force at P', a point just inside X and infinitesimally far from P. The direction of F_p is given by

$$\tan \delta = \frac{(F_{\frac{\pi}{2}L} + F'_{\frac{\pi}{2}L})}{(F_{oL} + F'_{oL})}$$

where δ is the clockwise angle that F_p makes with -x axis. The force at a point just inside W and infinitesimally far from P is equal and opposite to that at P'. Thus, the force direction has a discontinuity of $\pi/2$ at P but the absolute direction δ of the force relative to the edge direction, which is desired to be $\pi/2$, depends upon the parameters of the corner per the above equation.

B. Sensitivity to Intensity Model: There are two aspects of intensity model with regard to which the sensitivity could be evaluated: presence of noise in intensity values, and deviation from the homogeneity assumption we have made about region gray level. The first case is simple. If the regions contain independently distributed, zero-mean, additive noise, then the expected deviation in force at any point P is 0 compared to the case without noise. This is because changes in region intensities due to additive noise are spatially symmetric with respect to P, thus the deviation in attraction by one noisy pixel is cancelled on an average by another pixel on the opposite side of P. Therefore, the expected value of F_p is the same with or without noise. Second, if the region is not homogeneous in intensity then the force at a point in the region will depend upon the rate change of intensities away from P. For the simple case of an intensity ramp, the changes in intensity around P are still symmetric although nonzero as for the homogeneous case. Since force depends on the absolute intensity difference only, P still experiences equal and opposite forces from the up and down directions of the ramp resulting in a zero force as for the homogeneous case. This continues to hold for a ramp containing independent, zero-mean, additive noise for the same reason as for noisy homogeneous regions. For regions characterized by more complex distributions, e.g., polynomial variation in intensity, F_p may show direction discontinuities which reflect the structure. Such detected structure must be computed using Eqn. (2) for the specific case at hand. We have verified this edge preserving property of the transform about a corner (details in next Section).

C. Multiscale Behavior: Variation of scale parameters σ_s and σ_g leads to stable direction pattern of F as was discussed for the family of transforms in the previous section. These parameters change from point to point and therefore further processing is necessary to obtain a multiscale description of the image structure. This further processing is beyond the scope of the current paper, therefore we will not address it any further

for the specific transformed proposed.

5 Multiscale Shape Description

There are two ways in which the proposed approach yields multiscale description of region shape along with multiscale edges. First, we have seen that the detected edges form closed contours surrounding the corresponding regions. Therefore, multiscale edge detection implies detection of multiscale regions. The second way in which the transform extracts multiscale region shape information is by making explicit their skeletons. Recall from Property 1 above that in the vicinity of an edge point the magnitude of F decays away from region boundary and its direction points away from the boundary point. This implies that for the appropriate values of the scale parameters at each point, as one moves away from region boundary towards the interior there is a curve across which the force changes direction, from facing one side of the region boundary to another. If scale parameters not specific to the different points are used, this and other properties may not be observed. For example, if a value of σ_s which is too small is used inside a region, then the force may become 0 because of the homogeneity throughout the neighborhood over which the transform value significantly depends.

6 Experimental Results

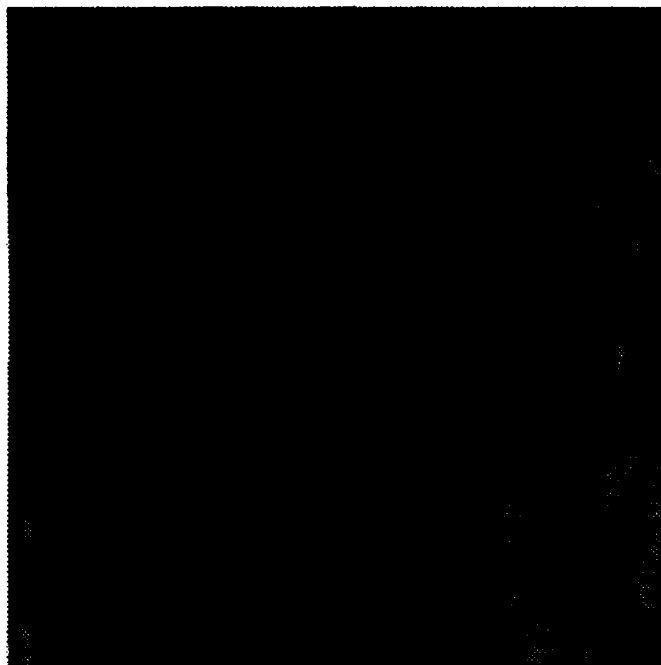
This section describes the experiments we have done to test the performance of the proposed transform. Since the combination of scale information from multiple pixels is outside the scope of this paper, the experiments were carried out to demonstrate the various properties of the transform, instead of obtaining edges or skeletons for images. To demonstrate each property, we have chosen appropriate parameters of the transform. These parameters would be selected automatically in the subsequent processing as outlined earlier. Our experiments demonstrated that the transform is fairly insensitive to the choice of the decay function. The results are rather similar for different choices, including step, linear, exponential, and Gaussian functions. Some of these functions were used to produce the results shown. This empirically confirms the power of the inherent directional sensitivity aspect of the transform, and our argument that the usual linear, convolution based edge detection has basic limitations.

Figure 5 shows the direction of the transform for an image, detected using fixed values of σ_s and σ_g at all pixels. Different directions of F are shown coded by different gray levels. This coding does lead to artifact edges when the directions corresponding to the dark and bright gray levels occur at adjacent image locations. The corresponding discontinuities should be ignored. Edges and region skeletons can be seen as directional discontinuities. The extracted structure would improve after the subsequent automatic processing of the transformed image not done in this paper. Fig 6 demonstrates the sensitivity of the transform to edge

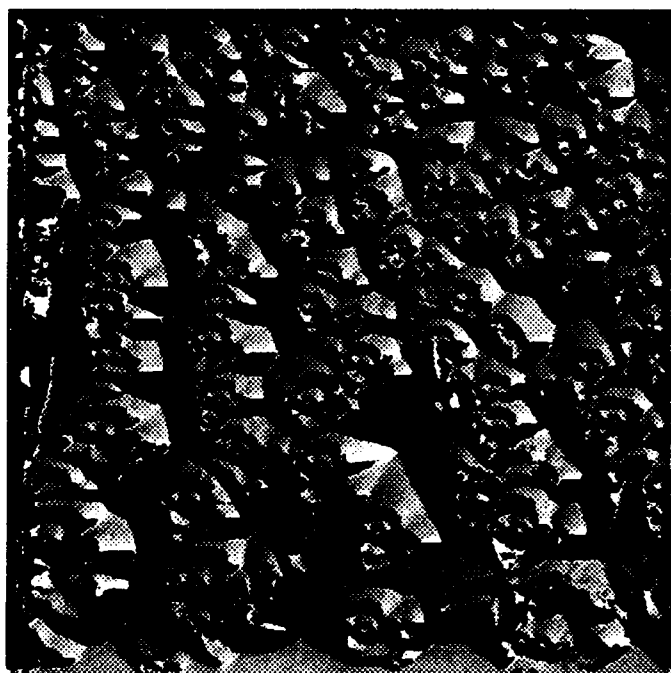
geometry. The synthetic image was designed to contain linear and curved segments as well as corners formed by them. The directional discontinuities coincide with edges for all parts of boundary without any smoothing. The skeletons of the regions are also clear from the directional discontinuities present along them. Fig. 7 shows the directions of F for the corner image of Fig. 4. This image was used in Sec. 4 to analytically evaluate the sensitivity of the transform to edge geometry by deriving closed form expressions for directional discontinuities along the edges. The experiments were conducted with corners of angles 15, 30, 45 and 90 degrees, with clean (binary) images as well as after adding zero-mean Gaussian noise having standard deviations of 10 and 30. In all cases, the edges were mostly preserved as there is a large directional discontinuity across edges. The magnitudes of the directional discontinuities created away from the edges due to noise are comparatively very small and easily distinguishable from those along the edges. Fig. 7 shows F for only two of the cases for brevity, including the sharpest corner (15 degrees). The vectors are shown as line segments at each point in the vicinity of the corner. The length of the line segment represents the vector magnitude and the tail of the vector is indicated by the end having a small filled circle. Such corners present a challenge to the common methods of edge detection. However, the results of the transform show that the directional discontinuities coincide with even the edges of the 15 degree corner without any rounding. Notice that the magnitude of the force away from the boundary decreases as expected and hence it is more susceptible to noise even though the expected response value is still the correct one. This will not happen after processing of the transform results with automatically chosen scale parameters since the value of σ , would be large towards the region interior, giving larger force magnitude and thus eliminating the increased susceptibility to noise.

References

- [1] N. Ahuja. A transform for detection of multiscale image structure. Technical Report CV-92-10-1, Beckman Institute, Univ. of Illinois, 1992.
- [2] V. Berzins. Accuracy of lapacian edge detectors. *Computer Vision, Graphics and Image Processing*, pages 195-210, 1984.
- [3] T. Binford. Inferring surfaces from images. *Artificial Intelligence*, pages 205-244.
- [4] J.F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 679-698, 1986.
- [5] L.S. Davis. A survey of edge detection techniques. *Computer Graphics and Image Processing*, pages 248-270, 1975.
- [6] R. M. Haralick. Digital step edges from zerocrossings of second directional derivative. *IEEE Trans-*



(a)



(b)

Figure 5. (a) A gray level image and (b) the force direction computed from the transform; the brightness at each pixel is proportional to the direction angle. When the directions corresponding to the dark and bright gray levels occur on adjacent image locations, artifact edges are visible which should be ignored.

- actions on *Pattern Analysis and Machine Intelligence*, pages 58–68, 1984.
- [7] S.L. Horowitz and T. Pavlidis. Picture segmentation by a directed split and merge procedure. In *Proc. Second Intl. Joint Conf. on Pattern Recognition*, pages 424–433, 1974.
 - [8] Y. Lu and R.C. Jain. Behavior of edges in scale space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 337–356, 1989.
 - [9] J. Malik and P. Perona. Edge detection by diffusion. Technical report, University of California, Berkeley, 1986.
 - [10] D. Marr and E. Hildreth. A theory of edge detection. In *Royal Society of London*, volume B-207, 1980.
 - [11] V. Nalwa and T. Binford. On detecting edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 699–714, 1986.
 - [12] P. Perona and J. Malik. Detecting and localizing edges composed of steps, peaks and roofs. In *Third Intl. Conf. Computer Vision*, pages 52–57, Osaka, Japan, Dec. 1990.
 - [13] J. Ponce and M. Brady. Toward a surface primal sketch. Technical Report MIT-AI-TR-824, MIT, 1985.
 - [14] A. Rosenfeld and A. Kak. *Digital Picture Processing*, volume 2. Academic Press, 1981.
 - [15] A. Witkin. Scale space filtering. In *International Joint Conference on Artificial Intelligence*, 1983.
 - [16] A. Yuille and T. Poggio. Scaling theorems for zero crossings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 15–25, 1986.

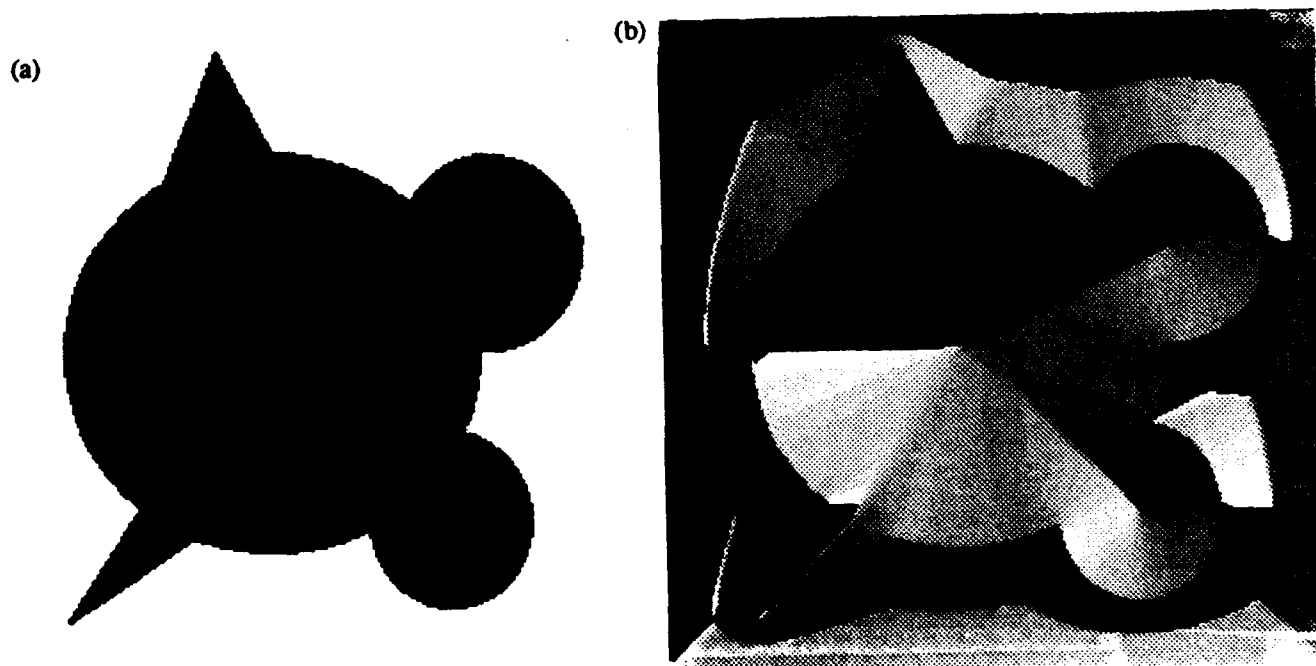


Figure 6. The sensitivity of the transform to edge geometry. (a) A binary synthetic image designed to test the performance of the transform near edges of different curvatures and their junctions. (b) The force directions computed by the transform shown as in Fig. 5. The edges and skeletons are visible as directional discontinuities.

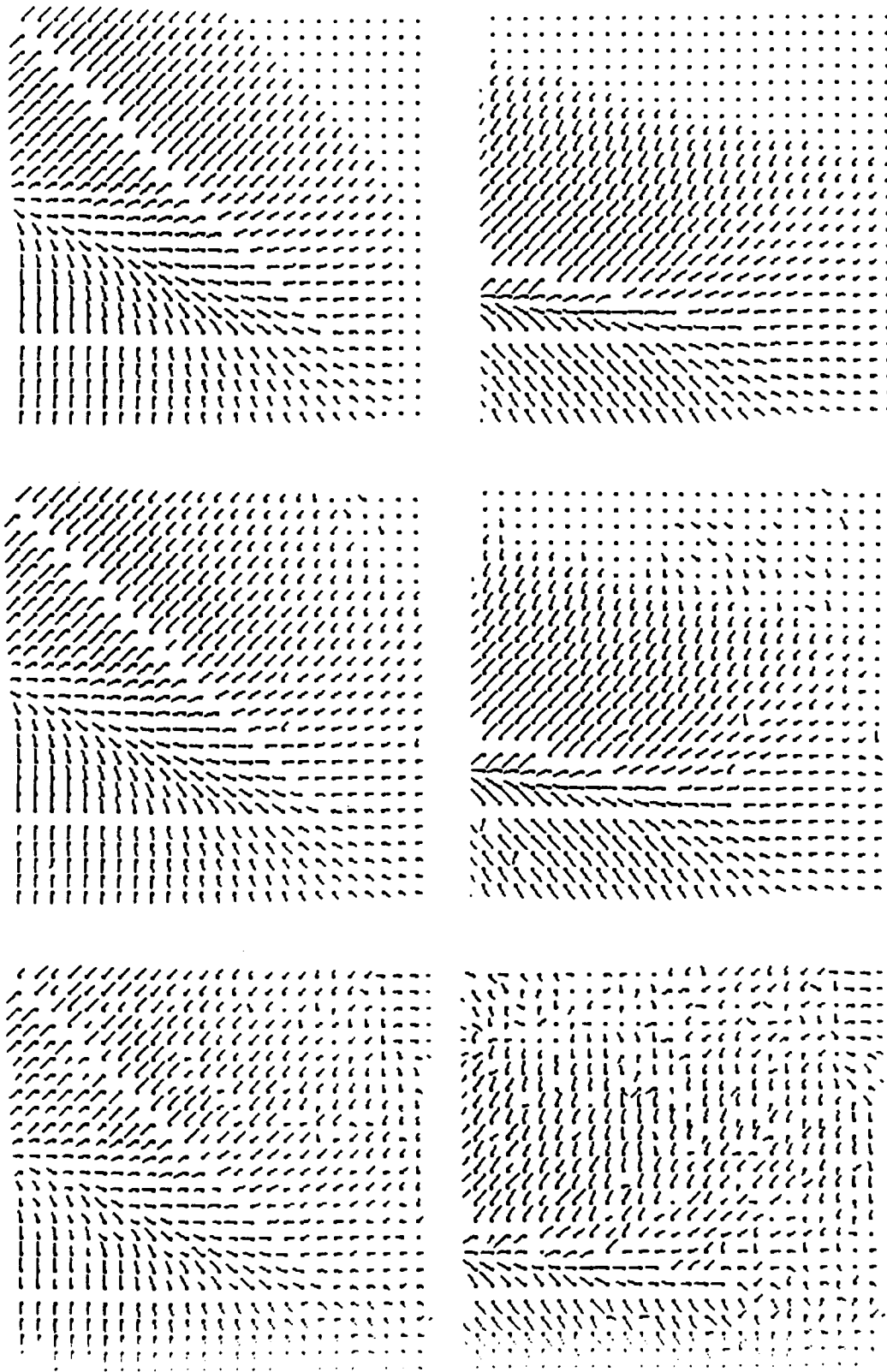


Figure 7. The force directions near a corner shown as vectors with length proportional to magnitude and the tail of the vector shown by a filled circle. Left column shows directions for a 45-degree corner with: (top) no noise, (middle) zero-mean, Gaussian additive noise having standard deviation 10, (bottom) same as (b) but standard deviation 30. Right column shows analogous results but for a 15 degree corner.

Scene Segmentation and Volumetric Descriptions of SHGCs from a Single Intensity Image

Mourad Zerroug and Ramakant Nevatia*
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles CA 90089-0273

Abstract

We present an approach for solving the figure-ground problem and computing volumetric descriptions in complex real images for an important class of objects: straight homogeneous generalized cylinders. Past work on shape description and recovery from contours assumed that perfect contours are given or that the scene has already been segmented. We address the problems of scene segmentation and shape recovery together. Our method is based on exploiting mathematical invariant properties of the contours of generalized cylinders in a multi-level perceptual grouping approach. The method handles SHGCs in complex scenes with occlusion and markings. We demonstrate the application of our method on complex real images. We also demonstrate the usage of the obtained descriptions for recovery of complete 3-D object centered descriptions of viewed objects.

1 Introduction

One of the fundamental problems in computer vision is the recovery of the shape of viewed objects in a scene from a monocular intensity image. A basic source of difficulty is the ambiguity caused by projection. Human vision does show that substantial information can be inferred from a single intensity image. This includes both segmentation of the scene into different objects and perception of their 3-D shape. To achieve such ability in machines has continued to be one of the most challenging problems in computer vision.

Among the cues that can be used for shape recovery from an intensity image, contour is the richest in geometric information and most robust to changes in viewing conditions. Using contour as a cue for shape description and recovery has received the attention of many researchers since the early days of the field. How-

ever, most previous work on inferring 3-D shape from 2-D contours assumes that the problem of object (surface) segmentation has been solved, whereas this is a key and difficult step in monocular scene analysis. Real images produce contours with many imperfections such as distortions, breaks and occlusion. Further, not just 'real' image contours are present in an image. Surface markings, shadows and noise also produce contours. Figure 1 shows an example of a real image and its extracted edges (by a Canny edge detector [Canny 1986]). The difficulty in dealing with such imperfections is that it is impossible, by just looking at the contours individually, to tell which constitute real contours and of what objects and which do not, or simply how many objects there are in the scene. This problem is known, in psychology, as the *figure-ground* problem and is more difficult in the presence of occlusion as substantial object contours may be invisible. To address this problem, it is necessary to use a grouping process to collect relevant features together and discard the irrelevant ones.

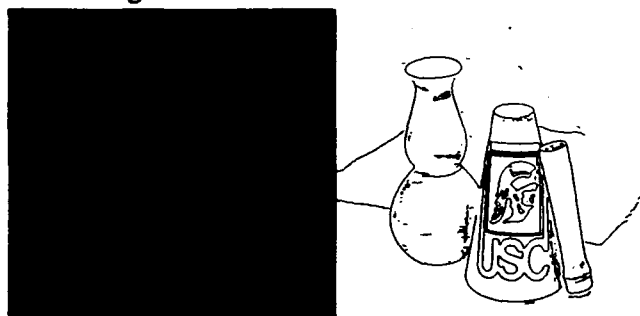


Figure 1 A real image and its extracted contours

Given the hierarchical nature of object and scene features, it is important to address the figure ground problem by using feature grouping at different levels of this hierarchy. In order to be of interest for imperfect contours, the grouping constraints should be locally applicable so as to handle occlusion and gaps. Yet, they should provide global criteria that discriminate between true instances and accidental ones.

In this paper, we describe our approach to the figure ground problem based on these observations. It addresses the problem of shape description and scene segmen-

* This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-90-C-0078. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

tation in the presence of multiple objects, broken contours, markings and occlusion for SHGCs, an important subclass of GCs [Binford 1971]. SHGCs do capture a large number of objects such as industrial parts and tools. Our method exploits *projective invariant properties* of SHGCs in order to guide the segmentation and description processes. We believe that invariant properties of generic shapes greatly help in solving the figure-ground problem. In fact, an essential characteristic of the segmentation of an image of a 3D scene should be its viewpoint invariance. The method we propose consists of a bottom up, perceptual grouping approach handling a hierarchy of three levels of features: curves, symmetries and SHGC surface patches. Throughout this hierarchy, the invariants are used for local feature detection, grouping of those features into consistent global ones and completion of missing features using only information from the image itself.

We have implemented a system that produced satisfactory results on rather complex scenes by the standards of currently developed methods. We have used these results for 3D shape recovery and we believe that they also have application in recognition.

The remainder of this paper is organized as follows. In section 2, we discuss related previous work. In section 3, we discuss the projective invariant properties of SHGCs we use. In section 4 we give an overview of our approach. Sections 5 and 6 discuss in detail the upper two levels of our hierarchy. Examples of obtained results are also given. Section 7 includes a discussion of robustness issues and a comparison of our method with others proposed in the literature. In section 8, we demonstrate the usage of those results for 3-D shape recovery. We conclude this paper in section 9.

2 Previous Work

Previous methods on generic shape detection and recovery can be classified into two broad classes: those assuming perfect data and those using real images.

Methods of shape recovery from perfect data focus on the recovery of 3D shape and assume that the problem of image segmentation has been solved. The method of Ulupinar and Nevatia [Ulupinar & Nevatia 1990a and b, Ulupinar & Nevatia 1991] addresses 3D recovery of certain classes of surfaces such as zero gaussian curvature surfaces, SHGCs and planar right constant generalized cylinders. Their method is based on the observation that certain types of symmetries provide strong constraints on 3D shape. The method of Gross and Boulton [Gross & Boulton 1990] addresses 3D recovery of SHGCs using a combination of constraints from contour and intensity information.

Methods of shape from real images explicitly address the problems of real image imperfections. They can be classified into two classes: specific model based and generic model based, although there is no clear dividing

line between them. An example of a specific model based approach is Acronym [Brooks 1983] which uses stored models to predict and match image features. That system has been used for the detection of airplane models (straight GCs) from a top view. Generic model based methods use 2D properties of generic 3D shapes, without using specific objects as models. Ponce *et al.* [Ponce *et al.* 1989] have derived projective invariants of the contours of SHGCs and used them in a simple method to detect their axes. Richetin *et al.* [Richetin *et al.* 1991] also used properties of the contours of SHGCs for pose estimation from an image of a single object. These last two methods do not address the segmentation problem. Rao and Nevatia [Rao & Nevatia 1989] and Mohan and Nevatia [Mohan & Nevatia 1989] have proposed two different methods for the figure-ground problem in complex scenes with occlusion, in the context of ribbons. Both methods address the problem of selecting symmetries and ribbons, a necessary task for the figure-ground problem. Those methods use rather intuitive constraints. Sato and Binford [Sato & Binford 1992a and b] have recently proposed a method to detect SHGCs. Their method and ours are quite similar in the principle of using projective properties. However, they differ in the way those properties are used and in the complexity of the scenes they can handle. Most notably, their system does not handle occlusion. We will compare it to ours in more detail in section 7.

3 Properties of SHGCs

Projective invariant properties provide strong constraints for the detection of generic shapes. Thus, detection of contours satisfying those constraints is an important step of the figure-ground problem and ensures that image segmentation is itself a viewpoint invariant process. Here, we include relevant properties from previous work and new properties we have derived. First, we give the definition of an SHGC.

Definition 1: An SHGC [Shafer & Kanade 1983] (straight homogeneous generalized cylinder) is the surface obtained by sweeping a planar cross-section curve C along a straight axis A while scaling it by a function r .

Let $C(t) = (u(t), v(t))$ be a parametrization of C , $r(s)$ the scaling function and α the angle between the cross-section plane and the SHGC axis (s -direction), then the surface of the SHGC can be parameterized as follows (using the formulation of [Shafer & Kanade 1983]):

$$S(t, s) = (u(t)r(s)\sin\alpha, v(t)r(s), s + u(t)r(s)\cos\alpha) \quad (1)$$

When $\alpha = \pi/2$, we obtain a *right* SHGC (RSHGC). Figure 1 shows an RSHGC and the chosen configuration of the axes. For an LSHGC the scaling function is linear, i.e. $r(s) = a(s - s_0)$. Curves of constant t are called *meridians* and curves of constant s are called *cross-sections* (also *parallels*).

In [Ulupinar & Nevatia 1990a], a class of symmetry

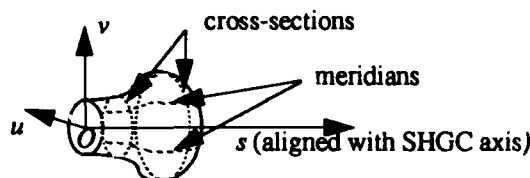


Figure 2 SHGC coordinate system and terminology called parallel symmetry is defined that is present in SHGC contours under certain conditions (discussed later). Its definition is given here.

Definition 2: Two planar unit speed curves¹ $C_1(w_1)$ and $C_2(w_2)$ are said to be parallel symmetric if there exists a continuous and monotonic function f , such that $T_1(w_1) = T_2(f(w_1))$. Where $T_i(w_i)$ is the unit tangent vector of $C_i(w_i)$. Thus, corresponding points have parallel tangent vectors.

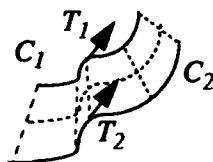


Figure 3 Example of parallel symmetric curves

The correspondence is said to be linear if f is a linear function. In this case the two curves are similar up to scale and translation. The axis is the locus of midpoints of lines of symmetry (correspondence lines). Figure 3 gives an example. A property of linear parallel symmetric curves is that lines of correspondences are either mutually parallel (for a unit scaling) or all intersect at one point (apex).

Now we state projective invariant properties of SHGCs. Some have been derived in previous work [Ponce et al. 1989, Shafer & Kanade 1983, Ulupinar & Nevatia 1990a]; we state those without proofs. Others are introduced here; proofs of these are contained in the appendix. Figure 4 illustrates those properties.

Property P1: Cross-section curves of an SHGC are mutually parallel symmetric with a linear correspondence. This property holds in 3-D and in the 2-D (orthographic) projection. The proof can be found in theorem 4 and its corollary in [Ulupinar & Nevatia 1990a].

Property P2: Contour generators (limbs) of an LSHGC are straight (they are meridians). This property holds also for the 2-D projection of limbs which are projections of those meridians. Therefore, in 2-D, the tangent line and any correspondence line at each limb point are colinear. The proof can be found in Section 4 of [Shafer & Kanade 1983].

Property P3: In 3-D, tangents to the surface in the direction of the meridians at points on the same cross-section, when not parallel, intersect at a common point on the axis of the SHGC [Shafer & Kanade 1983]. In 2-D, tangents to the projections of limbs intersect on the pro-

jection of the axis at a common point [Ponce et al. 1989, Ulupinar & Nevatia 1990a].

The properties we add have been reported, without proofs, in an overview of this work in [Nevatia et al. 1992]. Equivalent ones have been independently derived by [Sato & Binford 1992a and b]. Here, we state the new properties and give their proofs.

Property P4: We give this property in the form of a theorem and its corollary.

Theorem P4: Lines of correspondence between any pair of cross-section curves are either parallel to the axis or intersect on the axis at the same point. The proof of this theorem is given in appendix A.1.1.

Corollary P4: In 2-D (orthographic projection), lines of parallel symmetry between any pair of projected cross-sections are either parallel to the projection of the axis or intersect on it at a common point. The proof of this corollary is given in appendix A.1.2.

Property P5: Let $C_1(u)$ and $C_2(v)$ be two unit speed parallel symmetric curves with a linear correspondence $f(u) = au + b$. Then for all u and u' the vectors $V_1 = C_1(u') - C_1(u)$ and $V_2 = C_2(au' + b) - C_2(au + b)$ are parallel and $|V_2|/|V_1| = a$ (i.e. the ratio of their lengths is constant and equal to the scaling of the correspondence). The proof is given in appendix A.1.3.

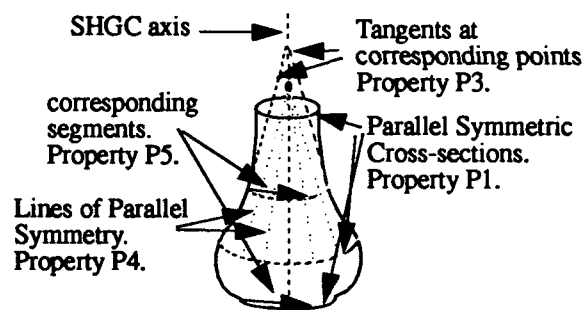


Figure 4 Projective invariant properties of SHGCs

The usage of these properties will be discussed throughout the description of the method in the next sections.

4 Overview of the approach

Our method operates at three perceptual grouping levels: the curve level, the parallel symmetry level and the SHGC patch level. The curve level grouping is aimed at forming contours from edges detected in a real image. The input to that level is an edge image and the output are global contours. Contours are first formed by an edge linking process based on simple contiguity criteria. Those contours are then segmented at corners. Obtained contours are usually discontinuous. At this level, a conservative co-curvilinearity based process is used for bridging short breaks. A method similar to [Mohan & Nevatia 1989] is used here, which consists of using an

¹a curve is unit speed if it is parameterized by arc length.

energy function measuring zeroth order and first order continuity between curve ends.

The parallel symmetry level grouping is aimed at forming global linear parallel symmetries between contours formed at the curve level. A hypothesize-verify process of several steps is used for that purpose. Linear parallel symmetries are used to hypothesize cross-sections of SHGCs.

The SHGC patch level grouping is aimed at forming global SHGC surface descriptions whenever possible. Another hypothesize-verify process is used that starts by detecting local SHGC patches (defined later) and generating grouping hypotheses of those patches and produces verified global SHGC hypotheses.

The perceptual grouping nature of our method allows to handle discontinuities in descriptions caused by occlusion and large gaps. The constraints for detection of features (parallel symmetries, SHGC patches) and grouping of those features are based on projective invariant properties discussed in section 3. Those properties are also used for completion of missing features such as broken contours and surfaces.

For lack of space, we will not discuss the curve level grouping. In section 4, we discuss the parallel symmetry level and in section 5, the SHGC patch level.

5 Parallel Symmetry Level Grouping

To detect parallel symmetries we use a hypothesize-verify process of several steps, from detection of local correspondences to forming consistent global ones. The steps are discussed below.

5.1 Detection of local correspondences

Local parallel symmetry correspondences are detected using the method of [Saint-Marc & Medioni 1990]. This method consists, first, of fitting quadratic B-splines to curves then finding correspondences analytically. The correspondences obtained are generally noisy, sparse due to breaks and may involve both desired and undesired symmetries (involving markings, for example). Further, the correspondences may not be linear, which is a requirement for cross-sections of SHGCs (property P1). Figure 5 gives some examples of correspondences given by that method. The second example has of the order of a thousand such symmetries. Grouping and selection of relevant correspondences are the objective of the next steps.

5.2 Grouping of parallel symmetries

The purpose of this step is to generate grouping hypotheses (connections) between symmetry elements. For this, we use a grouping method based on a local compatibility constraint derived from property P5. As shown in Figure 6, two symmetry elements ps_1 and ps_2 are considered for grouping if the vectors \vec{s} and \vec{r} defined by the end points of the symmetries are parallel and the ratio of their lengths $|\vec{s}|/|\vec{r}|$ (local scale) is

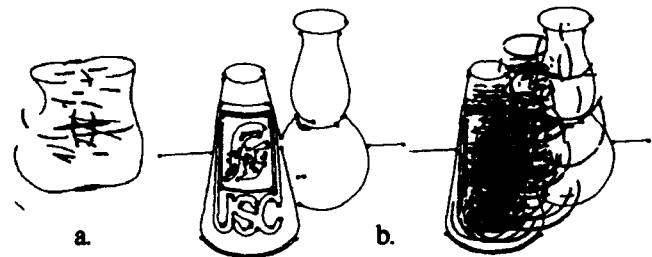
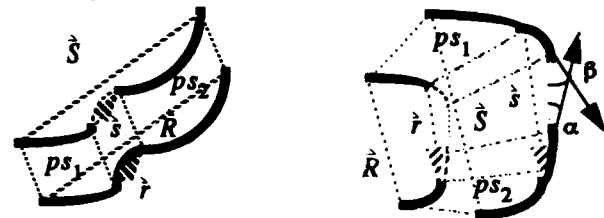


Figure 5 Local parallel symmetry correspondences (axes are in thick lines)

similar to the scale $|\vec{s}|/|\vec{r}|$ suggested by the grouped symmetries (global scale). In practice, we also use a connection measure for each grouping hypothesis. For this, we distinguish two cases, continuous connection and discontinuous connection.

1) Continuous connection: in this case the two symmetry elements have a common curve (Figure 6.a). The connection measure is based on the relative gap, $E = |\vec{s}|/|\vec{r}|$, between the non connected curves. A grouping hypothesis is generated if E is less than a fixed threshold. This measure has been introduced so as to penalize distant symmetry elements, a case that might occur between unrelated symmetries (involving markings, for example).



a. continuous connection b. discontinuous connection

$$E = |\vec{s}|/|\vec{r}|$$

$$E = (|\vec{s}|/|\vec{r}|)(\alpha^2 + \beta^2)$$

$$\text{local-scales} = |\vec{s}|/|\vec{r}| \quad \text{global-scales} = |\vec{s}|/|\vec{R}|$$

Figure 6 Grouping of parallel symmetry elements

2) Discontinuous connection: in this case, the symmetry elements do not share a common curve. Another connection measure is used in this case, $E = (|\vec{s}|/|\vec{r}|)(\alpha^2 + \beta^2)$, and is as shown in Figure 6.b. It controls both the gap and the continuity of the symmetric curves. Gaps that involve change in curvature sign are not considered.

This local compatibility constraint prevents grouping of symmetry elements such as the ones of Figure 7.a and Figure 7.b. Notice that grouping of symmetry elements implies grouping of curves involved in the symmetries. Therefore, this step implicitly handles (cross-section) curve groupings that have not been generated at the curve level due to large gaps or non smooth connections.

5.3 Selection of hypotheses

The previous step may produce a large number of

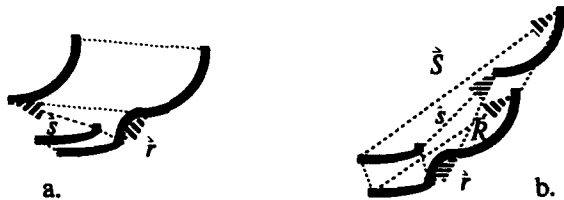
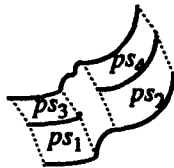


Figure 7 Non grouped symmetries. a. non parallel connections. b. non similar local and global scales.

connection hypotheses, some of them conflicting. Conflicts arise when there is more than one connection hypothesis involving the same curve at the same end. Figure 8 gives an example. At this level, it is difficult to decide which, among competing hypotheses, is the right one. Consequently, all alternative combinations involving non competing hypotheses (conflict free sets) are investigated.



the gap in the middle curve is differently influenced by the upper and lower curves

Figure 8 Competing hypotheses. Grouping of ps_1 and ps_2 competes with that of ps_3 and ps_4 .

5.4 Verification of global correspondences

In this step, hypothesized connections are checked for *geometric consistency*. The objective is to retain only those groupings that produce global parallel symmetries with *linear* correspondences. The verification consists of checking the similarity between the scale given by the global correspondence and the scales of each of its component parallel symmetry elements and connections (property P5). This is necessary because the local compatibility constraint only ensures scale similarity of two neighboring local correspondences².

5.5 Boundary completion

Selected global correspondences are used in order to fill in the gaps. Since symmetries are similarity relationships, missing boundaries of a curve can be inferred from corresponding boundaries of a symmetric curve. Boundary completion is different for the two types of connections discussed in 5.2. We discuss them separately.

1) **Continuous connection:** in this case, the common curve of the connected symmetry elements is used as a model for the missing boundary of the connection. This is done as follows.

- 1) the part of the model curve that corresponds to the gap is detected. For this, the global scale is used.
- 2) the missing boundary is obtained by scaling and translating the previous part so that it fills the gap.

²due to the usage of similarity measures, the relation may not be transitive.

This is shown by the dashed curve in Figure 6.a. This operation is done efficiently by the use of B-splines. The cross-section gaps in Figure 9.a and b have been so completed.

2) **Discontinuous connection:** in this case, there are gaps on both sides of the connection. The completion is done in two steps.

- 1) boundaries are inferred up to the extremities of the continuing curves (dashed curves in Figure 6.b). The same procedure as the one discussed previously is used here.
- 2) the two remaining gaps are filled in each by a quadratic B-spline (dotted curves in Figure 6.b).

The two filled in boundaries are parallel symmetric, thus producing a consistent boundary completion with the symmetry requirements. Figure 9.c is an example of such completion.

Finally, only symmetries involving closed curves are selected. Closure is defined by the existence of a cycle of curves connecting both extremities of a given curve. Gaps between adjacent curves in the cycle are completed by B-splines. This method has produced satisfactory results for all tested examples. Closed curves involved in parallel symmetries are likely to correspond to cross-sections of SHGCs (Property P1).

Figure 9 shows the results obtained in this level on some objects. Figure 9.d shows the completed cross-section of the cone of Figure 5.b.

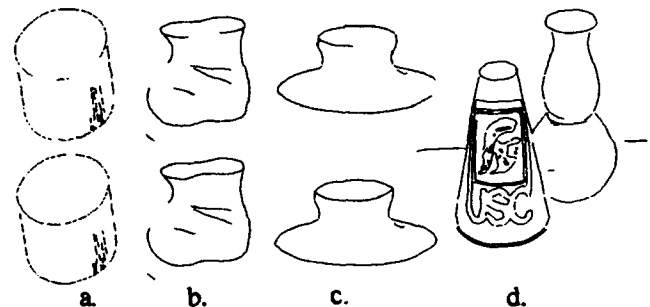


Figure 9 Results obtained in level 2. Original contours and completed correspondences.

6 SHGC Patch Level

At this level, the objective is to produce complete SHGC descriptions. Due to large gaps, usually caused by occlusion, a single surface may not be detected simply by searching for closed contours, or by expecting connectivity between surface extremities as in [Sato & Binford 1992a]. Furthermore, junctions and corners may not be reliable as cues for surface segmentation in a real image, as those features are themselves difficult to detect and sensitive to image imperfections. However, even under heavy occlusion, local surface patches, or surface segments, can still be detected. A fragmented surface can then be recovered by *grouping* those surface patches whenever there is evidence that they project

from the same global surface. In this level, we use a hypothesize-verify process of several steps in order to detect local SHGC patches and generate grouping hypotheses of these patches. The constraints used in this process are based on projective invariant properties discussed in section 3.

6.1 Detection of local SHGC patches

Definition 3: A *local SHGC patch* is given by a hypothesized closed cross-section and a pair of corresponding limb curves (*limb patches*) satisfying the projective properties P2 or P4; i.e. the limb patches are either straight (for a local LSHGC) or have the property that lines of symmetry between any pair of projected cross-sections intersect on a straight line (projection of the axis). Figure 10 shows sample local SHGC patches.

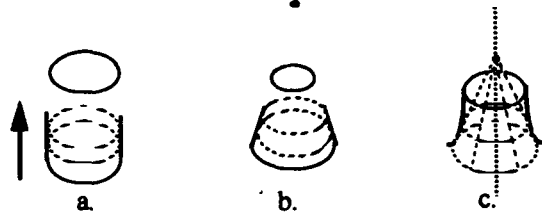


Figure 10 Sample local SHGC patches. a. cylindrical patch. b. conical patch. c. non-linear patch

For each hypothesized cross-section, the method consists of finding limb curves having such a correspondence. Curve segments lying between the two curves of a parallel symmetry (involving the hypothesized cross-section) are classified in two sets lying on the two sides of the parallel symmetry, say "left" and "right" side. For each pair of such candidate limbs we check whether they form corresponding limb patches. Using the given cross-section, the correspondence can be found using a method that minimizes the scale of the cross-section³ joining corresponding points [Ulupinar & Nevatia 1990a] (call it *limb based cross-section recovery*). Pairs of candidate limbs having such a correspondence are checked whether they form local SHGC patches:

- If the limb patches are straight, then a local LSHGC patch is hypothesized. The patch is further classified as being *cylindrical* if the two limb segments are parallel, or otherwise *conical*. In the first case, the limbs give an estimate of the direction of the axis (corollary P4). In the second case, the intersection of the lines supporting the limbs of a conical patch gives the cone apex, which belongs to the axis (also corollary P4). Cross-section recovery in this case is simple since all limb correspondence segments are parallel (property P5).
- If the limb patches are not both straight then corresponding points between the limb patches are identified using the limb-based recovery method. This

yields a set of recovered cross-sections (see Figure 11). Between *each pair* of such cross-sections having different scales, the intersection point of lines of symmetry is determined (Figure 10.c). A local SHGC patch is hypothesized if the locus of such points is a straight line (using fitting criteria). This line is a local estimate of the projection of the axis (corollary P4). We call this patch a *non-linear SHGC patch*.

We believe this method of finding axis points to be more robust and accurate than the method, used by Ponce *et al.* [Ponce *et al.* 1989], based on tangent lines (property P3), as the latter is sensitive to distortions of the limbs. In a sense, property P3 can be thought of as the limit case of corollary P4 where the two cross-sections get arbitrarily close to each other. The error in the slope of a line of correspondence is inversely proportional to the distance between the cross-sections. Thus, small errors in the tangent line direction greatly affect the position of the intersection (axis) points. Further, our method can be applied to $O(n^2)$ cross-sections (all combinations), providing more points for the voting process than the $O(n)$ property P3 would.

6.2 Grouping of local SHGC patches

The previous step may generate sparse local patches not all corresponding to perceived objects, as surface markings and contours from different objects may produce false hypotheses. Further, due to breaks and occlusion, several local patches can be obtained for the same (global) SHGC. In this step, grouping hypotheses are generated so as to form global patches describing complete primitive SHGCs. A combination of local geometric and structural compatibility constraints is used for hypothesis generation.

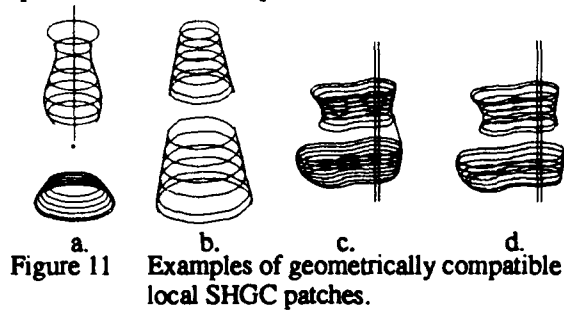
Geometric compatibility between two patches is based on corollary P4 and property P2. Depending on the types of the patches, several cases can occur:

- *non-linear* and *non-linear*: the axes must be (almost) colinear (Figure 11.c and d)
- *non-linear* and *conical*: the cone apex must lie on the axis (up to some error; Figure 11.a)
- *non-linear* and *cylindrical*: the direction of the cylinder must be (almost) parallel to the axis
- *conical* and *conical*: either the limbs are colinear (same apex as in Figure 11.b), or a line is generated (between to the apexes)⁴
- *conical* and *cylindrical*: a line is generated (apex and direction)
- *cylindrical* and *cylindrical*: the limbs must be colinear (for the same LSHGC), otherwise the directions must be parallel

³the scale is with respect to the hypothesized (completed) cross-section (which we will henceforth call "top" cross-section)

⁴the line will be constrained to be colinear to the global SHGC axis

Figure 11 shows some examples of geometrically compatible local SHGC patches.



Structural compatibility involves measures of proximity and continuity between SHGC patches. We distinguish two cases: *continuous connection* where the patches share a limb curve segment as for the SHGC in Figure 11.c, and *discontinuous connection* where there is no common limb as in Figure 11.d. A connection hypothesis is generated between an SHGC patch and a geometrically compatible neighbor if the connection is continuous or is discontinuous with the constraint that the limb extremities are co-curvilinear or form self-occlusion⁵. The co-curvilinearity measure uses looser thresholds than at the curve level since more information about the contours is given at this more global level.

Note finally, that a grouping of local SHGC patches implies a grouping of their limb curves. Therefore, gaps that have not been bridged in the curve level using co-curvilinearity may be bridged at this more global level.

6.3 Selection of hypotheses

Because of the highly constrained nature of the compatibility measures, conflicting hypotheses are rare at this level. When they do happen, they involve alternative connections at the same extremity of some local SHGC patch. The only selection done at this step consists of preferring continuous connections over discontinuous ones. Among the remaining hypotheses, the one involving the closest connection is selected.

6.4 Verification

In this step, grouping hypotheses (candidate global SHGCs) generated in the previous steps are checked for global consistency. Global consistency is checked by first determining the global axis of each set of connected patches, then verifying its compatibility with the axis of each of those patches.

The global axis detection depends on the nature of the local patches. If all the local patches are cylindrical with mutually colinear limbs, then they form a global cylindrical LSHGC with an axis direction determined from the direction of the global limbs. Similarly, if all the local patches are conical with colinear limbs, then they

form a global conical LSHGC whose apex is the intersection of the global limbs. Otherwise, the global axis (line) is detected by combining the recovered cross-sections of *all* component local SHGC patches in the same way discussed for the non-linear patches in section 6.1 (i.e. using corollary P4) and fitting a line to the obtained axis points. Figure 12 illustrates this procedure.

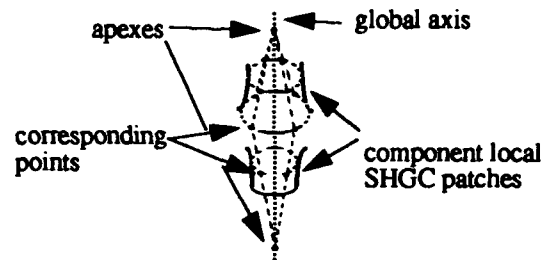


Figure 12 Global axis detection

Verification between the global axis and the component local ones uses the rules described for geometric compatibility in section 6.2. Successfully verified groupings form a global SHGC with a more accurate estimate of the axis.

6.5 Boundary completion

Global SHGC patches formed in the previous step consist only of aggregates of local patches believed to make up a single global surface. The descriptions of those global surfaces may thus be discontinuous if they are occluded or simply bounded by broken contours. This can be seen in the case of the occluded vase of Figure 9d, for which the surface boundary does not terminate due to that occlusion (another example is the case of the objects in Figure 11c and d). However, our (human) perception of a surface is clear there. In fact, we can even guess the shape of the hidden boundary due to the symmetric nature of the shape. We show that projective invariants can also be used for completion of surface descriptions. In this step, gaps in descriptions of verified global SHGCs are completed. Boundary completion is done for connections between adjacent local SHGC patches and *terminations* where a local SHGC patch, at an extremity of the global SHGC, has an incomplete limb correspondence due, say, to occlusion as is the case for the lower part of the occluded vase in Figure 9.d. The method consists of, first, recovering cross-sections at all points of the unmatched limb patch, using a method we call *axis-based cross-section recovery*, then finding the (missing) corresponding limb patch using a method we call *limb reconstruction method*. We discuss the two methods separately.

Axis-based cross-section recovery

Given the axis of the global SHGC and a reference cross-section (of any of the local patches), this method recovers cross-sections for unmatched limb patches (at connections or terminations). First, for each point P_u of a given unmatched limb, its corresponding point R_u on

⁵self-occlusion can be detected by T-junctions and discontinuities in cross-section scaling between local SHGC patches.

the reference cross-section is found (they have parallel tangents). See Figure 13. The scale of the recovered cross-section relatively to the reference one is determined as follows:

In the case of an LSHGC (Figure 13.a), the corresponding point P_c is simply the intersection of the line from P_u parallel to the limb correspondence line of the reference cross-section (line $R_u - R_c$ in the figure⁶) and the other straight limb of the LSHGC. The scale is given by the ratio $\text{dist}(P_u, P_c) / \text{dist}(R_u, R_c)$ (property P5).

In the case of a non-linear SHGC (Figure 13.b), let P_x be the intersection of the line connecting P_u to R_u and the axis⁷. Then, by the property of linear parallel symmetry between the cross-sections, it can be shown that the scale is given by the ratio $\text{dist}(P_u, P_x) / \text{dist}(R_u, P_x)$.



Figure 13 Axis based cross-section recovery method.
a. for LSHGCs. b. for nonlinear SHGCs

For a termination, the method is applied until the end of the unmatched limb is reached or there is overlap between the recovered cross-section and the *bottom end* cross-section of the SHGC. In doing so, we obtain a more accurate segmentation of limbs from cross-section. Figure 14.a shows the cross-sections so recovered for the connection of the SHGC of Figure 9.b and the termination of the SHGC of Figure 9.d.

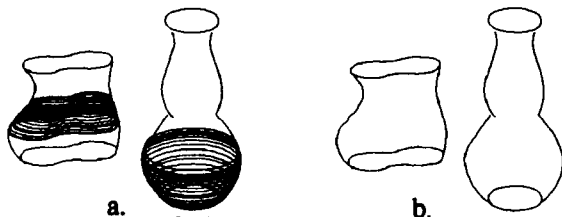


Figure 14 Cross-section recovery and limb reconstruction for previous SHGCs.

For discontinuous connections where there are no limb patches on either side of a connection, no cross-sections can be recovered. This leaves *holes* in the final description (see third SHGC in Figure 16a and b for which discontinuity is caused by self occlusion). In the case of LSHGCs, however, the recovery is straightforward as the limbs are known on both sides.

⁶ R_c is the point on the top cross-section whose tangent is parallel to the other straight limb.

⁷the reference cross-section is chosen so that the correspondence line is not parallel to the axis.

Limb reconstruction method

Cross-sections recovered by the previous method for a connection or a termination can be used to infer the missing limb boundary. The limb reconstruction method finds a point on each of the recovered cross-sections that is a limb point (in the projection of an SHGC, limbs and internal cross-sections are tangential to each other). The method consists of finding the *tangential envelope* of the set of recovered cross-sections. Given a starting point, call it P_0 as in Figure 15, taken to be an extremity at the *open side* of the connection (or termination), the method consists of finding a point P_1 on the first recovered cross-section whose tangent line passes through P_0 . P_1 is marked as a limb point. The process is then repeated for P_1 and the next cross-section until the limb point on the last cross-section is so determined. Since the axis-based cross-section recovery produces a dense set of very close cross-sections, the method essentially treats the infinitesimal patch between two successive cross-sections as being an LSHGC patch (where line tangents and limbs are colinear; i.e. property P2).

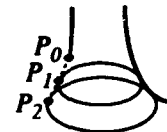


Figure 15 Limb reconstruction method.

For LSHGCs the limb reconstruction is straightforward as it only consists of extending the known straight limbs for continuous or discontinuous connections (or terminations). For discontinuous connections of a non linear SHGC, the boundary completion is not performed as no cross-sections could be recovered for the holes mentioned previously. However, the SHGC can still be used for 3-D shape recovery and recognition (the hole region will be left unspecified). Figure 14.b shows the limb boundaries so completed for the SHGCs of Figure 14.a.

Completed SHGCs are further verified for closure. Closure verification consists of checking required junction properties for the global SHGCs. Using the terminology used in [Malik 1987], limb curves and the *top* cross-section generally form *three-tangent* junctions⁸. At the *bottom* cross-section, they form *curvature-L* junctions. Because junctions cannot be expected to be perfect in real image contours, we use measures based on proximity and angular variations at junction points. For lack of space, we omit the details of this process. Hypotheses with non closed objects are rejected.

Results of this level are given in Figure 16. Figure 16.a shows the detected global SHGCs whose original contours have been given in Figure 9, except the last example whose original contours are given in

⁸in the case of edges (non limb boundaries) arrow and Y junctions are formed.

Figure 1 (for this latter, notice the completion of the occluded vase and cone). Recovered cross-sections and axes are shown. Figure 16.b shows the ruled surfaces (recovered cross-sections and meridians).

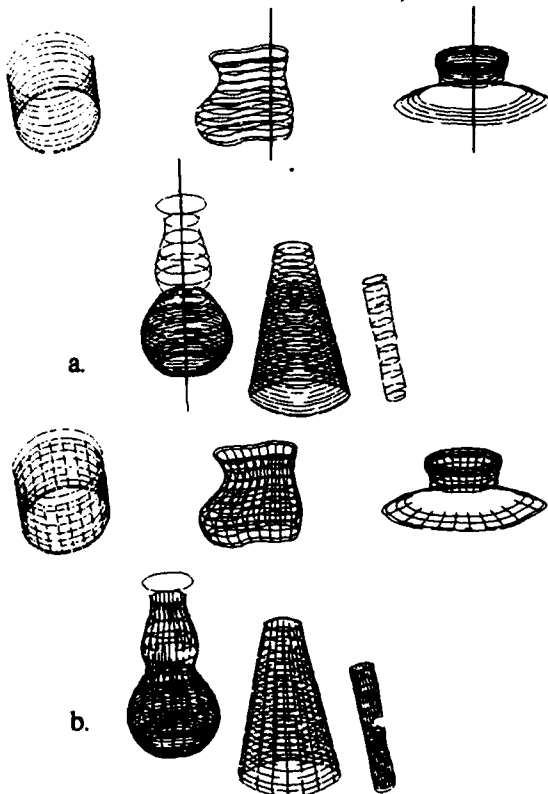


Figure 16 Results obtained in level 3. a. Obtained global SHGCs. b. Corresponding ruled surfaces. The last example is from the image in Figure 1.

7 Discussion and Comparison

We have tested our method on several images, about ten including variations of the one in Figure 1, and satisfactory results have been obtained. Four such examples are shown in Figure 16. Some of the simple examples have been presented here so as to illustrate the different steps of the method. The others involve multiple objects in different arrangements such as the last example in Figure 16. Also, a number of parameters have been used in the implementation of our system. We have mentioned them throughout the description of the method. For example, connection measures in the parallel symmetry level, linearity of limbs and axes and junction measures in the SHGC patch level. In all the tested images, the values of all parameters have been constant. Robustness of the system to changes in those parameters has been tested by changing their values by 50% of their default ones. Those changes have only affected the number of hypotheses and consequently the size of the search space. Looser thresholds produce larger search spaces. In the case of the contours of Figure 1, for example, increasing the values of the connection measures of the parallel symmetry level by 50% of their default

ones produced 17 connection hypotheses compared to 4 for the default values. Also, changes by 50% of the linearity thresholds, in the SHGC patch level, produced 95 local SHGC patches compared to 94 for the default values. Most importantly, the same final results have been obtained by changing the values of the parameters; i.e. one grouping hypothesis and two terminations accepted (selecting 4 local SHGC patches from the 94 originally hypothesized), resulting in the last three SHGCs given in Figure 16.

By way of comparison, the method of Sato and Binford [Sato & Binford 1992a and b] is similar to ours in the principle of using projective invariants to detect SHGCs. It differs, however, in two ways. First, application of the projective properties in their method is somewhat restricted to surfaces of revolution (SORs) and LSHGCs. Their application of the properties to surface detection, for example, may not give accurate results for general SHGCs as limb projections are generally not meridian projections. For example, in their system, correspondence segments between limbs are assumed to be parallel, whereas it can be shown that this is the case only for SORs and LSHGCs. We also believe that our application of the properties is more robust. For example, their parallel symmetry detection uses a Hough transform to detect the point of intersection of correspondence lines between symmetric curves. For symmetries with a scaling factor close to 1, the error in the detected point can be large. We use property 5 which is more robust to errors in correspondences as such errors cause only small changes in length ratios and directions of corresponding segments. The main difference between their method and ours, however, lies in handling occlusion and large gaps. The authors note that their system does not handle occlusion as simple connectivity criteria are used for surface detection. Our SHGC patch level grouping handles breaks and occlusion by detecting visible local surface patches and grouping of compatible ones into global surface descriptions. Further, our method also detects non-visible parts of the objects and completes them adequately thereby producing complete surface descriptions.

8 3-D Shape Recovery

We have applied the method of [Ulupinar & Nevatia 1990a] on our results. That method produces a viewer centered description, not the whole object description. However, it is often desirable to recover a complete 3-D object centered description of viewed objects also providing volumetric information. For this, we propose a method that uses constraints from both our 2-D description and the viewer centered description mentioned above. The method assumes that we have a Right SHGC (i.e. $\alpha = \pi/2$ in equation (1)). This equation becomes:

$$S(t, s) = (r(s)u(t), r(s)v(t) \cdot s) \quad (2)$$

Without loss of generality, we assume that $s = 0$ for

the top cross-section and that $r(0) = 1$ (the scaling is relative to the top cross-section). Note that our 2-D description already provides the values $r(s_i)$ (the scaling function is an *orthographic invariant*). However, it does not provide the values s_i (call them *s-values*) necessary for a complete description of the sweeping function.

Figure 17 gives the configuration of the coordinate systems relevant to this analysis. We denote

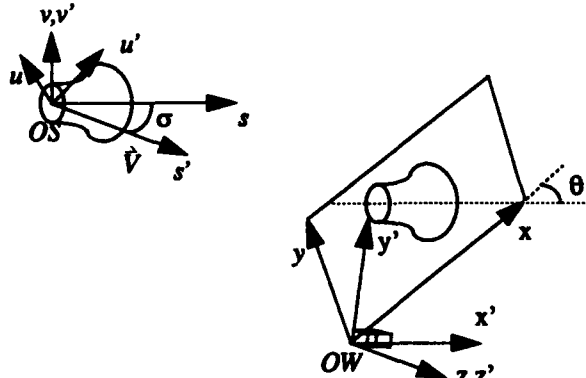


Figure 17 SHGC representation and projection geometry

$S = (OS, u, v, s)$ the SHGC coordinate system, $W = (OW, x, y, z)$ the viewer coordinate system, $I = (OW, x, y)$ the image coordinate system, \hat{V} the viewing direction (assumed, without loss of generality, to lie in the $u-s$ plane of S ; orthographic projection assumed). \hat{V} makes an angle σ with the s -axis (SHGC axis). Consider $S' = (OS, u', v', s')$ obtained by rotating S around v by σ so that the new s -axis (s') is aligned with \hat{V} . Let θ be the angle between the projection of the SHGC axis and the x -axis. Consider $W' = (OW, x', y', z')$ obtained by rotating W by θ around the z -axis so that the new x -axis (x') is parallel to the projection of the SHGC axis. Let $I' = (OW, x', y')$ be obtained by rotating I by θ . Then, in I' the SHGC axis is horizontal, and S' and W' differ only by a translation $OS-OW$. The relationship between coordinates in S, S', W' and I' are as follows (from now on we omit the arguments t and s in the expressions).

Letting (x_s, y_s, z_s) be the coordinates of OS in W' , a point P with coordinates (ru, rv, s) in S has coordinates $(\cos\sigma ru + s \sin\sigma, rv, -\sin\sigma ru + s \cos\sigma)$ in S' , $(\cos\sigma ru + s \sin\sigma + x_s, rv + y_s, -\sin\sigma ru + s \cos\sigma + z_s)$ in W' and its projection has coordinates $(\cos\sigma ru + s \sin\sigma + x_s, rv + y_s)$ in I' . In the remainder of this analysis, we consider image measurements in I' . World coordinates will be expressed directly in W' .

To recover a complete 3-D description of the SHGC, it is necessary to determine the 3-D top cross-section, the orientation of the axis, the coordinates (x_s, y_s, z_s) of its origin (point where the axis pierces the cross-section, not necessarily at its center) and the values s_i ($i = 1 \dots n$) (*s-values*) of the cross-sections (parallels) of interest. The recovery of those parameters is discussed in appendix A.2.

Application of this method to the descriptions obtained in Figure 16 is shown in Figure 18, where the objects are displayed for different values of slant and tilt from their original ones. (The right most object in the

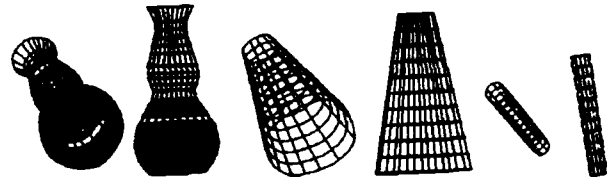


Figure 18 Recovered 3-D descriptions of previous SHGCs shown from different viewpoints.

figure has been interpreted as a LSHGC because the scaling function is close to be linear, producing straight limbs over its surface).

9 Conclusion

We have presented an approach to the figure-ground problem in real monocular images containing objects that can be described as SHGCs. It is based on two fundamental aspects. First, the study and derivation of projective invariant properties of SHGCs. Second, the multi-level perceptual grouping approach to scene segmentation and shape description. The projective invariant properties are the basis for detecting local correspondences, hypothesizing groupings, estimating more accurate global correspondences, verifying global consistency and completing missing boundaries. Thus, in a sense, our method not only filters out irrelevant features but detects where relevant ones are missing and completes them adequately. Our method differs from other perceptual grouping methods in that it uses rigorous constraints for the segmentation process as opposed to intuitive ones used by those methods. It also differs from other methods of invariants-based generic interpretation of image contours in that it uses a hierarchy of grouping levels handling substantial occlusion. We have also demonstrated the usage of our results to 3-D shape recovery, using a single image of a scene.

We plan on extending this approach to handling curved axis primitives (see [Zerroug & Nevatia 1993] for the derivation of *quasi-invariant* properties of curved axis GCs), and composite objects. These latter are made up of simple GC primitives. Detection of such objects can proceed by first detecting the component GCs and analyzing their structural and geometric relationships. Such objects introduce many difficulties that have not been addressed in this work, including incomplete cross-sections and non-parallel surface cuts of primitives, especially at joints. We believe, however, that the constraints and the methods developed here will be an important part of handling those objects.

Appendix

A.1 Proofs

A.1.1 Proof of theorem P4

Let $P_1 = S(t, s_1)$ and $P_2 = S(t, s_2)$ be two corresponding points on two different cross-sections. The line L joining these points can be parameterized as follows:

$$u = [u(t)\sin\alpha (r(s_1) - r(s_2))]m + u(t)\sin\alpha r(s_1) \quad (3)$$

$$v = [v(t) (r(s_1) - r(s_2))]m + v(t)r(s_1) \quad (4)$$

$$s = [s_1 - s_2 + u(t)\cos\alpha (r(s_1) - r(s_2))]m + s_1 + u(t)\cos\alpha r(s_1) \quad (5)$$

Case 1: $r(s_1) \neq r(s_2)$. The intersection point of L with the SHGC axis (s -axis) is given by setting $u = v = 0$; this yields $m = -r(s_1) / (r(s_1) - r(s_2))$ and the intersection point M has coordinates $(0, 0, (s_2 r(s_1) - s_1 r(s_2)) / (r(s_1) - r(s_2)))$ which are independent of t (i.e. of the particular points on the two cross-sections).

Case 2: $r(s_1) = r(s_2)$. In this case the direction of L is given by the vector $(0, 0, s_1 - s_2)$ which is independent of t and parallel to the axis \square .

A.1.2 Proof of corollary P4

An algebraic proof is not necessary as it is similar to the previous one, except that the expression of lines of correspondence is in the image plane. Instead, we give the following argument.

If the correspondence lines are parallel to the axis, then they are also parallel in the image (orthographic projection), otherwise the intersection property holds also true in the image since intersecting lines in 3D project onto intersecting lines in the 2D image (general viewpoint) \square .

A.1.3 Proof of property P5

$$\begin{aligned} C_1(u') - C_1(u) &= \int_u^{u'} T_1(s) ds = \int_u^{u'} T_2(as + b) ds \\ &= 1/a \int_v^{v'} T_2(t) dt = 1/a (C_2(v') - C_2(v)) \quad \square. \end{aligned}$$

A.2 Recovery of SHGC Parameters (3-D Object Centered Description)

A.2.1 Recovery of the 3-D cross-section curve

The method of [Ulupinar & Nevatia 1990a] determines the orientation $\vec{N} = (N_x, N_y, N_z)_{W'}$ of the cross-section plane. The back projection of the top cross-section points $(x_i, y_i)_I$ is thus given by $(x_i, y_i, z_i)_{W'}$, where $z_i = -(N_x x_i + N_y y_i) / N_z$ (the plane is arbitrarily fixed to pass through the origin of W).

A.2.2 Recovery of the viewing direction

From Figure 17, σ is the viewing angle (between the z -axis and the SHGC axis). The axis of the SHGC has its direction given by $-\vec{N}$. Thus $\cos\sigma = -N_z$ and $\sigma = \arccos(-N_z)$.

A.2.3 Recovery of LSHGCs

For LSHGCs $r(s) = (as + 1)$ in equation (2). For a cylinder, $a = 0$ (constant size cross-section). \vec{N} is the direc-

tion of sweep. It suffices to recover the s -value for the last ("bottom") cross-section. Let $P = (x, y)$ be a point on that cross-section and $P_0 = (x_0, y_0)$ its corresponding point on the "top" one. Then, using the relationship between coordinates of $\vec{C} = P - P_0$ in I' and S' , we obtain $s = (x - x_0) / \sin\sigma$.

For a cone ($a \neq 0$), we can determine a , the s -value of the last cross-section and the apex of the cone. Using the normal $\vec{n} = (n_x, n_y, n_z)$, determined by the method of [Ulupinar & Nevatia 1990a], at a point P (as above) and writing that \vec{n} and the meridian $\vec{C} = P - P_0$ are orthogonal (a LSHGC surface being developable, \vec{C} is in the tangent plane at P) yields:

$$s = -(\cos\sigma n_y (y - y_0) + (x - x_0) (\cos\sigma n_x - \sin\sigma n_z)) / n_z.$$

Using the value r of the scaling of the last cross-section (given by our 2-D description), a is given by $a = (r - 1) / s$. Writing that the apex of the cone, given by its 2-D coordinates (x_a, y_a) is the intersection point of all (straight) meridians and using the relationship between its coordinates in I' and S' yields the coordinates of the SHGC center $x_s = x_a - \sin\sigma (s(1 - r))$; $y_s = y_a$; $z_s = -(N_x x_s + N_y y_s) / N_z$.

A.2.4 Recovery of Non linear SHGCs

For each cross-section (parallel) C_j of interest, it is necessary to recover the value s_j along the axis. Our 2-D description already provides the values r_j (the scaling with respect to the top cross-section). Let P_j be a point on the surface of the SHGC (on any parallel). Let $\vec{m} = (m_x, m_y, m_z)$ be the tangent to the meridian ((m_x, m_y) can be computed in the image) and $\vec{n} = (n_x, n_y, n_z)$ the surface normal at P_j . Let $(r_j \mu, r_j \nu, s_j)$ be the coordinates of P_j in S , (x, y, z) its coordinates in W' ((x, y) are known from image measurements) and (x_0, y_0, z_0) the coordinates of P_0 , the corresponding point of P_j on the top cross-section ((x_0, y_0, z_0) are determined as discussed in A.2.1).

Writing that \vec{n} and \vec{m} are orthogonal (basic differential geometry) and equating expressions of the tangent to the meridian in S' and W' yields $u = v ((\cos\sigma m_x - \sin\sigma m_z) / m_y)$; where v can be computed with good accuracy using the relationship between coordinates of P_0 in S' and W' , by $v = y_0 - y_s$ (y_s is the constant second coordinate of the projection of the axis in I' ; recall that in that system the axis is horizontal). Equating expressions of the correspondence vector $\vec{C} = P_j - P_0$ in I' and S' yields $s_j = (x - x_0 - \cos\sigma (r_j - 1)) / (\sin\sigma)$.

In practice, the above process is performed on points where the tangents to the meridians are not parallel to the axis or orthogonal to it. More than one point is used for each cross-section. The obtained values are averaged to obtain an estimate of the value of s_j .

Let C_j be an arbitrary cross-section, r_j ($r_j \neq 1$) its scaling with respect to the top cross-section, and s_j its s -value computed by the method described previously. Let

(x_a, y_a) be the image coordinates of the intersection point of all lines of symmetry between C_j and the top cross-section ((x_a, y_a) are given by our 2-D description). From (2) and the analysis in appendix A.1.1, the 3-D coordinates of that point in S are $(0, 0, s_j / (1 - r_j))$. Equating these with their expression in W yields $x_s = x_a - (\sin \alpha s_j) / (1 - r_j)$; $y_s = y_a$; and $z_s = -(N_x x_s + N_y y_s) / N_z$.

References

- [Biederman 1987] I. Biederman, "Recognition by components: A theory of human image understanding," *Psychological Review*, 94(2):115-147, 1987.
- [Binford 1971] T.O. Binford, "Visual perception by computer," *IEEE Conference on Systems and Controls*, December 1971, Miami.
- [Binford 1981] T.O. Binford, "Inferring surfaces from images," *Artificial Intelligence*, 17:205-245, 1981.
- [Binford 1991] T.O. Binford, "Inverse generalized cylinders: Inverse generalized Transformational Invariance and Quasi-Invariance," In *Proceedings of Darpa-Esprit Workshop on Invariance in Computer Vision*, 1991.
- [Brooks 1983] R.A. Brooks, "Model-based three dimensional interpretation of two dimensional images," *IEEE Transactions PAMI*, 5(2):140-150, 1983.
- [Canny 1986] J.F. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 8(6):679-698, November 1986.
- [Clowes 1971] M. B. Clowes, "On Seeing Things," *Artificial Intelligence* 2, 1, 1971, pages. 76-116.
- [Gross & Boulton 1990] A. Gross and T. Boulton, "Recovery of generalized cylinders from a single intensity view," In *Proceedings of the Image Understanding Workshop*, pages 557-564, Pennsylvania, 1990.
- [Koenderink 1990] J.J. Koenderink, "Solid Shape," *M.I.T. Press*, Cambridge, MA, 1990.
- [Lowe 1985] D.G. Lowe, "Perceptual Organization and Visual Recognition," *Kluwer Academic Publishers*, Hingham, MA, 1985.
- [Malik 1987] J. Malik, "Interpreting line drawings of curved objects," *International Journal of Computer Vision*, 1(1):73-103, 1987.
- [Mohan & Nevatia 1989] R. Mohan and R. Nevatia, "Segmentation and Description of scenes Using Perceptual Organization". In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 333-341, 1989.
- [Nevatia & Binford 1977] R. Nevatia and T.O. Binford, "Description and recognition of complex curved objects," *Artificial Intelligence*, 8(1):77-98, 1977.
- [Nevatia et al. 1992] R. Nevatia, K. Price and G. Medioni, "USC image understanding research: 1990-1991," In *Proceedings of the Image Understanding Workshop*, San Diego, California, January 1992.
- [Ponce et al. 1989] J. Ponce, D. Chelberg and W.B. Mann, "Invariant properties of straight homogeneous generalized cylinders and their contours," *IEEE Transactions PAMI*, 11(9):951-966, 1989.
- [Rao & Medioni 1988] K. Rao and G. Medioni, "Useful geometric properties of the generalized cone," In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 276-281, 1988.
- [Rao & Nevatia 1989] K. Rao and R. Nevatia, "Description of complex objects from incomplete and imperfect data," In *Proceedings of the Image Understanding Workshop*, pages 399-414, Palo Alto, California, May 1989.
- [Richetin et al. 1991] M. Richetin, M. Dhome, J.T. Laprestre and G. Rives, "Inverse Perspective Transform Using Zero-Curvature Contours Points: Applications to the Localization of Some Generalized Cylinders from a Single View," *IEEE Transactions PAMI*, 13(2):185-192, 1991.
- [Saint-Marc & Medioni 1990] P. Saint-Marc and G. Medioni, "B-spline contour representation and symmetry detection," In *First European Conference on Computer Vision*, pages 604-606, Antibes, France, April 1990.
- [Sato & Binford 1992a] H. Sato and T.O. Binford, "On finding the ends of SHGCs in an edge image," In *Proceedings of the Image Understanding Workshop*, San Diego, California, January 1992.
- [Sato & Binford 1992b] H. Sato and T.O. Binford, "BUILDER-I: a system for the extraction of SHGC objects in an edge image," In *Proceedings of the Image Understanding Workshop*, San Diego, California, January 1992.
- [Shafer & Kanade 1983] S.A. Shafer and T. Kanade, "The theory of straight homogeneous generalized cylinders," *Technical Report CS-083-105*, Carnegie Mellon University, 1983.
- [Ulupinar & Nevatia 1990a] F. Ulupinar and R. Nevatia, "Shape from contours: SHGCs," In *Proceedings of IEEE International Conference on Computer Vision*, pages 582-582, Osaka, Japan, 1990.
- [Ulupinar & Nevatia 1990b] F. Ulupinar and R. Nevatia, "Inferring shape from contours for curved surfaces," In *Proceedings of International Conference on Pattern Recognition*, pages 147-154, Atlantic City, NJ, 1990.
- [Ulupinar & Nevatia 1991] F. Ulupinar and R. Nevatia, "Recovering Shape from Contour for Constant Cross Section Generalized Cylinders," In *Proceedings of Computer Vision and Pattern Recognition*, pages 674-676, 1991, Maui, Hawaii.
- [Zerroug & Nevatia 1993] M. Zerroug and R. Nevatia, "Quasi-invariant properties and 3D shape recovery of non-straight, non-constant generalized cylinders," In *Proceedings of the Image Understanding Workshop*, Washington DC., 1993.

Saliency Detection and Partitioning Planar Curves*

Martin A. Fischler and Helen C. Wolf

Artificial Intelligence Center

SRI International

333 Ravenswood Ave., Menlo Park, CA 94025

(fischler@ai.sri.com wolf@ai.sri.com)

Abstract

This paper summarizes the underlying ideas and algorithmic details of a computer program that performs at a human level of competence for a significant subset of the *curve partitioning* task. It extends and "rounds out" the technique and philosophical approach originally presented in a 1986 paper by Fischler and Bolles. In particular, it provides a unified strategy for selecting and dealing with interactions between salient points, even when these points are salient at "different scales of resolution." Experimental results are described involving on the order of 1000 real and synthetically generated images.

Index Terms: computer vision, salient points, critical points, curve partitioning, curve segmentation, curve description

1. Introduction

A critical problem in machine vision is how to break up (partition) the perceived world into coherent or meaningful parts prior to knowing the identity of these parts. Almost all current machine vision paradigms require some form of partitioning as an early simplification step to avoid having to resolve a combinatorially large number of alternatives in the subsequent analysis process. Given this critical role for partitioning as a functional requirement of a complete vision system, it is a major challenge to find some significant subset of the partitioning problem for which an algorithmic procedure can duplicate normal human performance. This paper (a compressed version of a much longer document which will appear in IEEE PAMI later this year) summarizes the underlying ideas and algorithmic details of a computer program which performs at a human level of competence for a significant subset of the *curve partitioning* task. It extends and "rounds out" the technique and philosophical approach originally presented in a 1986 PAMI paper by Fischler and Bolles [Fischler86]. For example, it provides a unified strategy for resolving conflicts in selecting

among neighboring potential partition points that may be salient at different "scales of resolution."

While our focus in this paper is on curve partitioning in a generalized setting (the curves in our experiments are mostly without semantic meaning), and where the criterion for success is duplicating normal human performance, finding salient points on image curves (potential partition points) plays a critical role in both two and three dimensional object recognition, in curve approximation, in tracking moving objects, and in many other tasks in machine vision.

In many approaches to 2-D object recognition, objects are represented by their boundaries, and the recognition techniques depend (directly or indirectly) on locating distinguished points along the boundary; typically these distinguished points are discontinuities or extrema of local curvature (sometimes called "corner points") and inflection points [e.g., Mokhtarian86]. "Corners" on the contours of imaged objects are often used as features for tracking the motion of these objects and for computing optical flow [e.g. Mehrotra90]. In 3-D recognition, partitioning is typically one of the first analysis steps – especially when objects can occlude each other. Hoffman and Richards [Hoffman82] argue that when 3-D parts are joined to create complex objects, concavities will generally be observed in their silhouettes, and that segmentation of image contours at concavities (the maxima of negative curvature along the contours) is a good strategy to decompose (even unmodeled) objects into their "natural parts."

In cartography, computer graphics, and scene analysis, it is often desirable to partition an extended boundary or a contour into a sequence of simply represented primitives (e.g., straight line segments or polynomial curves of some higher degree) to simplify subsequent analysis and to minimize storage requirements [e.g., Teh89].

In our own current work concerned with delineating linear structures in aerial images, the technique presented in this paper was an essential component of the system (briefly described in Appendix C) that produced the results displayed in Figure 6.

*This work was performed under contracts supported by the Defense Advanced Research Projects Agency.

2. Problem Statement

In its most general sense, partitioning involves assigning, to every element of a given "object" set, a label from a given "label" set. For our purposes in this paper, the object set is the set of points along a curve (or contour segment) lying in a prescribed region of a two-dimensional plane. While we deal with cases where the points in the object set do not form a continuous digital curve, in most of our exposition in this paper we will assume that the curves are continuous¹ and non-intersecting. Our label set is binary, points will be called either significant (critical) or non-significant, for some specified purpose. In Fischler and Bolles [Fischler86], it is demonstrated (or at least argued) that perceptual partitioning is *not* independent of some assumed task or purpose. In this paper we focus on one of the three tasks discussed in the above reference: Selecting a small number of points (called *critpts*) along a curve segment which could be used as the basis for *reconstructing* the curve at some future time. Figure 1 shows the specific instructions and curves used in one set of relevant experiments involving human subjects; this figure also shows the critpts that were selected by the subjects, and the comparable results produced by our algorithm (called the Saliency Selection System, or SSS, and discussed in Appendix B).

In order to separate the generic partitioning criteria used by human subjects from criteria based on their past experience, such as when the subject is able to assign a name to the curve (e.g., the curve looks like the letter "s"), we used "random" curve segments for our experiments; the technique employed to generate the segments is described in Appendix A. We also wanted to avoid having to deal with the recognition of global features (e.g., symmetry or repeated structure, or even straight lines and analytic curves) as a condition for making critpt selections; avoiding this problem is justified if we are correct in our belief that local and global analysis are accomplished by separate mechanisms. In order to deal with global features, the complexity of any solution would be expanded enormously since a whole new vocabulary of such features and their representations would have to be implemented. The generation and use of random curves took care of this problem also (i.e., it is highly unlikely that symmetries or repeated structure would ever be generated by our random process).

3. Relevance, Prior Work, and Critical Issues

The partitioning problem has been a subject of intense investigation since the earliest work began in ma-

¹Each point of the non-branching one pixel wide curve, with coordinates (x,y), has one or more neighbors with x-coordinates in the set (x+1, x, x-1), and y-coordinates in the set (y+1, y, y-1).

chine vision. It has been widely assumed that in order to reduce the combinatorics of scene analysis to a manageable level, it is necessary to decompose images into their meaningful component parts as one of the first steps in the analysis process. The difficulty arises from the need to partition the image into parts before we know the identity of those parts. The underlying assumption then is that there are generic criteria, independent of the goal of the analysis, that if discovered, could be used to obtain useful (or at least, intuitively acceptable) partitioning; additional problem dependent criteria could be always added to produce a more relevant result for some particular purpose.

The partitioning problem becomes progressively harder as we increase the number of dimensions in which we are working; in this paper we only address the 1.5-D problem of partitioning planar curves. A specific criterion which can form the basis of such partitioning was originally proposed by Attneave [Attneave54] - points at which the curve bends most sharply are good partition points.² This idea has been the starting point for most of the subsequent efforts in curve partitioning, but attempts to convert this abstract concept into a computationally executable procedure, that gives intuitively acceptable results, has met with limited success.³ References [Imai86, Mokhtarian86, Pavlidis74, Rosenfeld73, Teh89, Wuescher91] are representative of work in this area.⁴

The main problems we must solve are:

- (a) A way of assigning a measure (or degree) of saliency/criticality⁵ to each point on a curve. Most investigators have equated sharp bending of a curve with the mathematical concept of curvature, but curvature is not well-defined for a finite sequence of points (which is how our sensor acquired curves are generally represented). Further, it is not obvious that the mathematical definition of curvature is the best computational approximation to the human criteria for criticality. In Fischler and Bolles [Fischler86], bending is interpreted

²Hoffman and Richards [Hoffman82] give convincing evidence that we should distinguish between positive and negative curvature maxima. That is, on closed curves, extreme points of negative curvature - associated with object concavities - have greater utility as partition points than positive curvature maxima, but the positive maxima (and inflection points) play an important role in describing the individual segments.

³As noted later, most of the work on the curve partitioning problem, especially recent work, has *not* been concerned with duplicating generic human performance, but rather with performing specific visual tasks having different criteria for success.

⁴The approach taken by Wuescher and Boyer is distinct in that they first extract contour segments of approximately constant curvature and then infer the location of partition points as a secondary operation.

⁵We will use the terms *saliency* and *criticality* somewhat interchangeably in this paper. However, saliency can be considered to be the generic subset of points that are critical for some partitioning task.

as deviation from straightness – it is closely related to proposed approximations to mathematical curvature, as illustrated in Figures 2 and 3, but has a number of advantages: it is an easily measured quantity, even for digital curves (i.e., sequences of coordinate pairs), and as discussed in the next section, its local extrema are in better accord with human preference (choices based on approximations to the definition of mathematical curvature occasionally include anomalous points as shown in the examples of Figures 2 and 3).

- (b) A way of adjusting the criticality of a given curve-point to take into account its interactions with its neighbors; i.e., **local context**. It is obvious that human subjects will often avoid assigning a critpt label to both members of a pair of points, even when both points have high (independent) criticality values, if the points are close neighbors along the curve. The basic approach of local non-maximum suppression is not sufficient, in itself, to duplicate human performance.
- (c) A way of dealing with the interactions between critpts that are significant at **different scales of resolution**. If a human subject looks through a fixed sized window at the same curve segment displayed at two different magnifications, the selected critpts will not always be the same, and the selection at the lower resolution will not always be a subset of those at the higher resolution (e.g., Figure 4). This is in contrast to the commonly held assumption that critpt assignment should be independent of "scale of resolution."
- (d) A **threshold of significance**; a minimal level of criticality below which variations are considered to be noise and no critpt designations are made. (Some investigators reject the idea that any user supplied parameters or thresholds should be necessary.)

We have addressed the above issues through the solutions to a set of subproblems:

1. Definition of an algorithmic procedure (which is parameterized to deal with noise and scale) for assigning criticality values to each point on a curve independent of decisions made about the locations of (other) critpts. The solution to this problem, essentially the procedure given in Fischler and Bolles [Fischler86], provides answers at a human level of performance for isolated critpts (i.e., along a section of a random curve, generated as described in Appendix A, for which human subjects select only one critpt). Thus, for the domains we experimented with (and especially the domain defined in Appendix A), we were able to assign fixed values to scale/resolution and noise/significance parameters

so that our program would make the same selections as human subjects when there was near unanimous agreement among these subjects. This algorithm is described in Appendix B.

2. An analysis of how geometric scaling of the input curve, and resolution specific operations on the curve, can be equated, and thus the development of a basis for normalizing criticality scores across scale.
3. Development of a general approach to the problem of resolving the competition/cooperation interactions of geometrically related objects based on "local dominance." The same machinery used to deal with interactions at a given scale of resolution is also used to resolve conflicts across different scales of resolution.

In the remainder of this paper, we describe our solutions to the problems enumerated above, and then present examples and experimental results to justify the design decisions we made and to illustrate the performance capabilities of our algorithm.

4. Evaluation of Saliency

Saliency is a critical attribute (for description and recognition) assigned to perceived things in the world by the human visual system (HVS). While an elusive concept in general, task specific specializations of this concept are easily found that elicit consistent choices across human subjects. An acceptable computational definition of contour/curve saliency must provide ⁶

- The specification of a procedure that quantifies the abruptness and extent of the deviation of a curve from its straight-line continuation; a sharp bend is more salient than a shallow one, and the greater the excursion, the more prominent/salient the "feature."
- Agreement with human judgement in terms of both selection, and accuracy of placement, of the critical points (in some well defined context).

4.1 A Computational Definition of Saliency

Conventional definitions of curvature present a number of serious problems with respect to their use as a saliency measure in computational vision (CV). First, the mathematical definition is based on the properties of a curve in the infinitesimal neighborhood about the

⁶In this paper we are primarily concerned with saliency based on *local* cues; locations on a curve where there is a transition from one type of curvature behavior to another, e.g. from perfectly straight to "wiggley," may also be psychologically salient, but such forms of *global* saliency are beyond the scope of our current investigation.

point at which curvature is being measured. For the finite precision quantized curves dealt with in CV, it has been difficult to find a suitable approximation to the limiting process originally intended for use on *mathematically continuous* curves. Second, it is readily observed that saliency is not an infinitesimal point property, but is based on some finite extent of the curve. A proposed solution to both problems, offered by Rosenfeld and Johnston [Rosenfeld73] was to find an appropriately sized segment of the curve about the point in question, and take a "snapshot" of the limiting process at this single (implied) scale. That is, rather than the rate of change of tangent angle with respect to curve length, R/J proposed measuring the angle between two fixed length chords, where the lengths correspond to the computed "natural scale" of the curve about the given point. We will call this curvature-analog the R/J-Curvature. There are a number of other definitions of mathematical curvature (e.g., the limiting radius of a circle whose three defining points converge at the curve-point in question) which have analogs that could have been used in place of the angle measure in R/J-Curvature but these definitions are monotonically related, and do not really present distinct alternatives. Thus, R/J-Curvature is a suitable representative for the whole class of mathematical curvature-measure analogs.

In Fischler and Bolles [Fischler86], our concern was not to find a good digital analog for curvature, but rather to find an effective measure of saliency. The quantity defined in that paper can be viewed as a curvature-extremum measure in which the limiting process (in scale) is replaced by a scanning process (in space) more appropriate to digital curves. The scanning process is parameterized by scale, and the resulting measure is a signed quantity which we call F/B-Saliency (F/B-S).

While the particular choice of a curvature measure as a component in a complete system for selecting the most salient points (critpts) on a planar curve depends on many factors, it is still interesting to compare the raw scores returned by *curvature-analogs represented by the R/J-Curvature* with the extreme points (ultimately) selected by our algorithm (SSS) as shown in Figures 2 and 3 for a randomly generated curve. In these figures we observe problem situations that highlight some of the differences between the two underlying metrics (R/J-Curvature and F/B-Saliency).⁷

There are some problems with *any* raw measure of curvature that must be dealt with by using procedures that

invoke (at least) local *context*. For example, in Figure 3 we see a case (double arrow) where two critpts were selected at almost adjacent locations along the curve. This undesirable behavior was not eliminated by the simple "non-maximum suppression" filter that produced good results in most other situations. It is necessary to use more specific criteria in deciding when two critpts are too close together, and also, what to do when the adjacent points have equal saliency scores (e.g., arbitrarily eliminate one of them or eliminate both and place a new critpt between them). In Figure 3 we see cases (two single arrows) where almost invisible features were chosen as critpts because they *did* have locally extreme curvature scores; how do we decide when to reject such occurrences. In Figure 2 we see a case where a critpt (designated by an arrow) was inserted at a location displaced from the position we consider correct; this was due, in part, to the length of the arms of the angle measuring "operator" relative to the size of the feature (see Figure 2d) - it is not always possible (or practical) to find an appropriate operator size for every potential feature. In the following sections (and appendices) of this paper we describe and justify the methods we employ to deal with these problems. The issue we are primarily concerned with in this section is the choice of a basic saliency metric. We justify our preference for the F/B-S metric on two grounds:

1. Unlike the fixed scale mathematical (FSM) curvature analogs (e.g., R/J-curvature), F/B-S rarely makes an error in positioning a critpt, or in ignoring a salient point that human observers would select. The issue here is robustness, F/B-S integrates information over an extended set of "looks" at the curve segment containing the point whose saliency is being measured. FSM techniques take a single look at the situation. Thus, our main problem with the F/B-S metric is selecting the most salient of the selected critpts to be retained as our final result (the filtering operation generally involves the elimination of less than half of the points originally selected).
2. The F/B-S metric is responsive to both the curvature and the size of a curve "feature." This provides a common basis for ranking critpts at a given scale (so that the larger of two geometrically similar objects is assigned a higher saliency score) as well as across scales by taking into account the size of the operator. The FSM-curvature analogs are insensitive to the size of the feature - they inherit the mathematical property that curvature is a point property and only the smallest neighborhood about a point that allows us to measure curvature is relevant (this implies a single "natural scale" at any point on a curve; a concept we reject, e.g., see Figure 4).

⁷In both of the figures, we used fixed common scale parameters for both metrics as noted in the figure captions. It should be remembered that R/J-curvature, as we define it in this paper, is representative of a whole class of curvature-based metrics and is not intended to duplicate the complete Rosenfeld/Johnston algorithm - they also incorporate a procedure for finding a preferred stick length. However, many of the problems with the performance of the complete algorithm, which are discussed in Davis77 and in other of the papers we reference, can be observed in the performance of the R/J-Curvature metric.

4.2 Comparison of the Saliency Selection System (SSS) with Human Performance

The primary criterion for judging the competence of the overall saliency selection system (SSS) we present in this paper is its ability to match human performance – both in the defined task and with respect to generic evaluation of the selected critpts. We performed a set of informal experiments with 11 human subjects (also see the experiments described in Fischler86). The instructions given to the subjects and the resulting selections are shown in Figure 1. We also show the selections made by the SSS algorithm. The results of these (and additional but not described) experiments can be summarized as follows

- At least 9 of the 11 subjects selected the same set of six or more critpts on each of the four curves we used in the experiments, and the SSS chose the same set of critpts. Every critpt selected by the SSS was also selected by at least one human subject.
- In spite of the high degree of consistency in the overall selection of salient points, the human subjects differed in the order in which they chose these points. We tried a number of experiments in which the only difference was a very slight change in the wording of the instructions, and obtained different orderings (across the same set of selected points) from our subjects. It is obvious that the subjects used a global strategy to match the task (different for each subject) to choose the order in which the points were selected – even though the specific points selected were largely determined by local context.

In addition to the curves used in the human experiments, we ran the SSS algorithm on (the order of) 1000 randomly generated curves with no obvious errors. Figure 5 shows the results of a (typical) sequence of 40 consecutive experiments.

5. Dealing with the Problems of Scale and Resolution

A vision system, concerned with creating a description of some object that may be encountered again in the future, perhaps when the object is closer or further away, must take scale or magnification into account when deciding what shape elements to pay attention to. Under extreme changes in resolution, when salient features might appear or disappear, it may not be possible to make an informed judgement in the assignment of relative saliency scores; but for a limited range about a given resolution, this should indeed be possible.

Obviously, geometric properties of objects that are invariant over scale are especially valuable in describing and recognizing the objects, since absolute scale is often impossible to judge in an image, and even relative

scale can be difficult to describe or measure if the measurement must be referenced to the global geometry of the object. One of the main issues we address in this paper is how to define extrema in the "bending" of a curve as a local effectively scale-invariant property that is in agreement with the judgement of the human visual system.

If we define criticality of points on a digitally represented curve in terms of quantities that have dimensions that must be measured by some physical process, then there is no direct way of invoking such formally defined mathematical concepts as the derivative, or curvature, which require limiting processes of infinite resolution. Approximations to these concepts are resolution dependent (e.g., the size of the operator employed) and measurements made on most objects will not "scale" in any simple or uniform way. Further, if we examine a curve through a fixed size window (either a fixed region of a computer screen, or the foveal region of the human retina), and we successively increase the resolution at which the curve is displayed, some of its parts will eventually disappear from view, and some of the smaller original structures, that were not significant, will now dominate the visible appearance of the curve (e.g., Figure 4).

If the mathematical definition of curvature were applicable to digital imagery, then many (but not all) of the issues of scale could be resolved. There is still the problem that a very small "glitch" can have a very high value of curvature but a very low psychological significance. Thus the scale or size of a "feature" (e.g., the glitch) is an issue. The term "feature" does not appear in our problem definition; in fact, by focusing on local curve properties, we had hoped to eliminate the need to invoke this concept since an appropriate definition is far from obvious.⁸ Since scale can't be ignored (even if we had a good approximation for curvature in the digital domain that was independent of scale) the following questions arise:

- The distinction, if any, between resolution and scale
- How to choose a range of scales appropriate to the specified performance criteria
- How to measure criticality at different scales
- How to compare criticality values computed at different scales
- The relationship between smoothing and scale change

⁸Intuitively, there are sections of any given curve that we call features; these entities provide the psychological basis for the selection and relative saliency of the associated critpts. Critpts are markers that define the shape and boundary of features – the extent of the curve corresponding to a feature will generally subsume the "region of support" for the curvepoints comprising the feature. Features can overlap, and their boundaries are not always apparent.

- The relation between operator size and scale change
- How to make cooperation/competition judgements across scales
- How to determine the features for which we expect consistency (of criticality scores) to hold across scales, and where such consistency can't be expected (if the latter were never the case, we could always do our analysis at one scale and compute the criticality values at other scales as needed).

While consistency at all scales and for all features is not possible, over some range of scales (say 5:1) we expect there to be a "normalization" factor which allows us to compare the saliency scores computed at one scale with values computed at other scales. We would also expect that relative locations of local extrema for certain features would remain fixed as a curve is scaled, regardless of the size/scale of the operator that assigns the criticality scores.

Some of the earliest work (e.g., Rosenfeld and Johnston) on finding salient points merged the problem of assigning a curvature measure to a point with that of determining the scale at which to measure curvature. The key idea is that each point has a single scale at which its curvature should be measured - this scale is usually found by a search process over successively larger scales until some measured quantity achieves a local extremum.

5.1 Change of Scale Vs. Change of Resolution

If we magnify a continuous curve that was originally represented at infinite precision, every point of the new image corresponds to a point in the original image, but its x and y coordinate values have been multiplied by some real number which we will call the scale factor. No information was introduced nor lost, but the physical space required to render the curve has increased. However, if the original curve was represented at finite resolution (e.g., each point as a pair of integer coordinates), then (say) doubling the scale leaves us with a disconnected set of points. Filling in the gaps requires introducing new information. Here we will say that a change of resolution has occurred (a change in resolution can also result in the loss of information, as in the case of demagnification or smoothing at some fixed resolution). Thus, the concept of a scale change corresponds to a reversible transformation, while, in general, a change in resolution involves an irreversible process in which information is lost (as in smoothing), or new information is introduced (as can occur in zooming).

If we compute the curvature for points on a continuous (infinite resolution) curve at two different scales, we will generally get two distinct sets of values (e.g., a circle with radius 2 is a scaled version of a circle with radius 1, but by definition, their curvatures are in the ratio 1:2. On the other hand, the angles of a triangle remain

unaltered under a scale change). It will be the case, however, that for smooth curves, the local extrema will be found at corresponding locations - but even here, the numerical values of curvature will not scale in any simple way (curvature is a nonlinear function).

5.2 SSS Mechanisms for Evaluating Saliency at Different Scales and Resolutions

In designing a computational module to evaluate saliency subject to the ideas discussed above, we can pursue at least three distinct strategies:

1. Assume that saliency is independent of scale, or that there is a natural scale associated with each location on the curve that must be discovered.
2. Use a fixed scale saliency measure, but generate multiple versions of the given curve at some predetermined set of scales.
3. Parameterize the saliency measure to give results approximating those that would be obtained from strategy (2) for the selected scales.

We previously argued against strategy (1) on the assumption that a unique natural scale *cannot* generally be associated with a single curvepoint (see Figure 4). We have chosen strategy (3) since strategies (2) and (3) are conceptually compatible, but (3) could be computationally more efficient if we can find a simple way to use some combination of operator scaling and score normalization so that both approaches give (nominally) the same scores in most situations. Intuitively, doubling the stick length (in the F/B-S metric) for a simple convex section of a curve should result in four times the score assigned to the corresponding critpt: The stick is now positioned twice the distance from the critpt in most of its "looks" (i.e., placements of the stick which subsume a curve segment containing the critpt), and there are twice as many looks. Thus, the procedure we employ, *normalizing all scores by dividing by the square of the sticklength*, will leave invariant the saliency scores assigned to features which should be scale invariant, such as the angle formed by two (effectively) infinite straight lines. On the other hand, for those features that have limited extent along the curve, comparable to the scales we wish to discriminate among, the larger scaled versions of the features will be assigned higher scores.

6. Cooperation/Competition Interactions Between Critical Points

An important contribution of this paper over the work presented in Fischler and Bolles [Fischler86] is a major revision of the approach to filtering the critpts, based both on comparisons at a given scale as well as across

different scales. At a conceptual level, there are two main differences.

First, in the earlier work we did not use the information about the sign (concavity/convexity) of the computed F/B-Saliency; in our current algorithm, we separate all the candidate critpts into two sets corresponding to positive and negative F/B-S.⁹ These two sets are processed independently of each other (by identical procedures) and the resulting selections are combined by logical union to produce the final output. Our own observations confirm those of other researchers (e.g., Hoffman and Richards), that positive and negative curvature extrema appear to be distinguished from each other by the HVS, in part because they play different roles in partitioning and description tasks.

Second, in the earlier work we used a simple "dominance" criterion for competition of closely spaced critpts detected at different scales. A critpt detected at some given scale would suppress all critpts detected at smaller scales (shorter "sticklength") that were located within a specified scale related distance from it. This rule rarely produced "ugly" errors, but occasionally caused the obviously correct critpt to be deleted in favor of one slightly displaced from the preferred location. A significant portion of the work described in this paper has been focused on finding a more effective and uniform basis for establishing "local dominance." In other sections of this paper we provided a justification for a normalization factor which would permit us to assign a saliency ranking to competing critpts, regardless of the scale at which they were originally detected. Thus, competition, both within and across different scales is now treated in a uniform manner. In the following subsection we discuss some of the specific problems that must be resolved in competition resolution, and the algorithmic procedures we invoke to deal with these problems.

6.1 Mechanisms for Filtering Competing Critpts

One of the algorithmic mechanisms we devised to deal with the above problems (described in greater detail in Appendix B) is to construct an array with one slot for each indexed location along the curve (conceptually two such arrays, one each respectively for positive and negative saliency scores). Each slot is either free or "owned" by exactly one critpt. A critpt occupies only one of the slots it owns – this occupied slot corresponds to its actual location along the curve. A "new" critpt,¹⁰ contending for a slot, must have a normalized score greater than the

⁹For an open curve segment, the assignment of positive vs. negative is arbitrary; the important consideration is that we use the information about the direction of deviation of the curve from the stick to separate detected critpts into the two possible categories which are then processed separately.

¹⁰All the potential critpts are detected, sorted, and then entered into the array in increasing order of saliency to avoid sequence dependent effects.

existing value stored in the slot to capture it. If a new critpt captures a slot occupied by (as opposed to simply being *owned* by) a previously dominant critpt, all of the slots of the now dominated critpt are also captured. This mechanism provides a way of avoiding the need to choose a fixed-sized "base of support" for a critpt.

7. Algorithm Performance

The algorithm discussed in the previous sections of this paper, and described in Appendix B, has been compared with human performance (Figure 1), and has been run on hundreds of randomly generated images (as described in Appendix A) without making any obvious errors. In all these cases the same set of parameters were used with no operator involvement. Figure 5 shows 40 consecutively generated random curves and the critpts selected by the algorithm. Figure 6 in Appendix C shows results of the algorithm run on curves extracted from real images.

8. Discussion

Curve partitioning is an active research area which not only is of theoretical interest as a basic element in pictorial description (e.g., Attneave, Bengtsson and Eklundh, Hoffman and Richards), and for providing insight into the partitioning problem in general (e.g., Fischler and Bolles), but has many potential applications. Some of the more immediate ones include: data compression by using critpts as the basis for regenerating a curve by straight line or spline interpolation (e.g. Imai and Iri, Teh and Chin), matching/recognition using critpts and/or the partitioned curve segments (e.g., Mokhtarian and Mackworth, Wuescher and Boyer), and as a key component of an interface for man-machine communication about pictorial objects (the ability to point at icons representing symbolic objects has revolutionized the computer-user interface; to extend this capability, one would like to be able to point to a location in an image and have the machine be able to deduce the component being referred to – image partitioning in general, and especially curve partitioning, are critical to this goal).

In this paper we have focused on one specific aspect of the curve partitioning problem: Duplicating human performance in the selection of a small number of points (called *critpts*) along a curve segment which could be used as the basis for *reconstructing* the curve at some future time. While there will generally be a significant degree of overlap in the points selected by the techniques referenced above (focused on different applications), there are also significant differences. There has been very little recent work on the generic problem of choosing psychologically salient points with which to directly compare our results. On the other hand, we have conducted a relatively large number of experiments with

uniformly good results (e.g., see Figure 5).

There are two major paradigms¹¹ underlying the published work on partitioning planar curves. The first involves obtaining a mathematically differentiable representation of the given digital curve by the use of splining or Gaussian convolution (e.g., Mokhtarian86). This gives good results for many applications, but the salient points on the *smoothed* curve are often displaced from their original locations (or eliminated). This paradigm is not suitable for our purposes in this paper.

The second paradigm, which includes the work described here, is to first measure some approximation to the curvature at each point on a curve. This usually involves choosing, or finding, an appropriate scale at which to make the curvature measurement. This is typically accomplished by making the curvature measurement over increasingly larger curve segments (centered on the curve point being evaluated) until either the computed curvature at the point, or some related quantity, reaches a local extrema. Each point is assigned a saliency/criticality value (its estimated curvature) and an interval length along the curve centered on the point (called its *region of support*). The region of support is then used for non-maximum suppression – each point suppresses other points with lower criticality scores falling in its region of support.

Major differences between our approach and other work under this second paradigm include:

- A generic saliency measure which often selects points corresponding to local curvature extrema, but which in many situations is in better accord with human selection preference and placement accuracy.
- A distinct approach to the problem of dealing with curve *features* salient at different scales. The conventional approach is to associate a single scale with each curve point which in turn defines a fixed *region of support* to be used for non-maximum suppression. In our approach, we measure the saliency of each curve point at a number of different scales, and have developed procedures for allowing potential *critpts*, found at different scales and spatial locations to compete¹² with each other. This competition is not restricted to any fixed extent of the curve (which thus avoids anomalous selections caused by an important event occurring just beyond the fixed limit of search, i.e., the *horizon effect*).

¹¹Additional approaches are available for partitioning 1-D curves; for example, see Fischler and Wolf [Fischler83] or Witkin [Witkin83]. As noted in Appendix B, the 1-D partitioning technique in the Fischler83 reference is used as a component of the SSS algorithm.

¹²It is interesting to note that we have not found a use for cooperative reinforcement – cooperation appears to be a global relation. Competition is important at the local level (e.g., lateral inhibition)

Our approach to local saliency selection can be considered a form of automated preattentive perception. Potential extensions could include dealing with more global curve features, such as recognizing the intersection of extended straight line segments, or transition points between analytic curves with different parameters, or global symmetries and repeated structure. Recognizing these more global structures, and ranking them with respect to human perceived saliency, may well fall outside the competence of the basic approach described in this paper.

9. Appendices

9.1 Appendix A: Generation of Random Curves

The following method was used to construct the random curves used in the experiments described in the body of this paper.

(1) Thirty (x,y) pairs are generated for each curve. Each value of x and y are generated by a uniform-distribution (0-1) random-number generator and then multiplied by 100 to produce numbers (coordinate-values) uniformly distributed between 0 and 100.

(2) The thirty points are next linked by a minimal-spanning-tree (MST).

(3) A diameter path is extracted from the MST, and the ordered subset of the original randomly generated points that fall along this diameter path are the input sequence provided to a spline-fitting routine [Cline74] which returns a continuous curve represented by a sequence of (x,y) coordinate pairs. These sequences, typically containing on the order of 150-250 points, are the random curves used in our experiments.

9.2 Appendix B: An Algorithm For Computing Curve-Point Criticality

The partitioning algorithm described in Fischler and Bolles [Fischler86] has been modified and extended as summarized below.

The algorithm collects candidates (peaks) for the critical points of a curve by examining the deviation of the points of the curve from a chord or "stick" that is iteratively advanced along the curve. Sticks of different lengths are used to find critical points that are salient at different "natural" scales on the given curve. (Except when explicitly stated otherwise, two sticks were used for all the experiments discussed in this paper; one of length 10 pixels and the other of length 20 pixels.) The algorithm provides the option of using arc-length along the curve, or the euclidean length of the stick, to determine the separation of the endpoints of the stick on the curve; we used the euclidean length of the stick for all of the experiments discussed in this paper. One end of the stick is advanced along the curve, one pixel at a time, and the other end is placed at the first (sequential)

position further along the curve for which the Euclidean distance equals or exceeds the specified stick length.

For each placement of the stick, an accumulator associated with the curve-point (in the interval of the curve between the two endpoints of the stick) of maximum deviation from the stick is incremented by the absolute value of the distance from the point to the stick if this distance exceeds a predefined noise threshold. However, for the given stick placement, if there is more than one excursion (exit and return) outside the noise region, the underlying model is violated and the accumulators are not incremented. (The noise threshold was uniformly set to 20 percent of stick length; thus a euclidean deviation of more than 2 "pixels" from a stick of length 10 was required to cause any modification of the associated accumulator.)

To deal with direction dependent effects, a complete traverse is made in both directions along the curve summing the results in the same accumulators. The points which have locally maximum scores in the accumulators (called peaks) for any of a given set of sticks are the points from which the critical points will be selected.

The following information is collected for each peak and used to find the critical points:

- INDEX: the sequence number along the curve of the point at which the peak was located.
- STICK: the length of the stick (in pixels) used to find the peak.
- DEV: the sign of the deviation of the peak with respect to the curve.
- NSCORE: the "normalized" score which is the score in the accumulator for the peak divided by the square of the stick length.

The peaks are divided into two groups with like-signed deviation DEV. The critical points for the two groups are found independently of each other and their union is returned as the set of critical points for the curve.

In finding the critical points, we stipulate that each peak's score has a region of support, *plus* and *minus* half its associated stick length, on each side of its position along the curve. An array (the support array) equal to the length of the curve is used to store the support information. The support information for a peak is a list (NSCORE INDEX STICK). For each peak, the support information may be entered at every index location covered by the region of support depending on what was previously stored in the location.

For all locations in the support region for the new peak (in the support array), an entry at J is replaced by the information for the new peak if there is no previous entry in the array or if the score for the new peak is > than the score in the existing entry in the array. In addition, if the entry J is being replaced, and J is also

the INDEX for a peak that was entered previously, the support information for the new peak replaces the support information of the old peak wherever it occurs in the support array (i.e. even outside of the new peak's original support region).

After the above processing, the critical points for the curve are designated as those points whose index into the support array equals the index stored in the information list of the array element.

It can be seen that the order in which peaks are entered into the support array can affect the final selection of the critical points because a peak's region of support can be altered by the "capture" process, and thus depends on the state of the support array at the time the peak is entered. In our implementation of the algorithm for running the experiments, we entered the peaks into the support array as soon as they were computed in order to gain computational efficiency and simplicity, and still obtained excellent results. In the current version of the algorithm we collect all the peaks for all the sticks, sort the peaks by their normalized scores, and then enter them into the support array in order of increasing score.

There are some additional aspects of the algorithm that are further discussed in the more complete version of this paper, including ways to handle problems associated with very sharp angles and competing critpts of approximately equal saliency scores,

9.3 Appendix C: Partitioning Curves Extracted From Aerial Imagery

A technique for detecting and delineating low resolution linear structures appearing in aerial imagery, such as roads and rivers, was described by the authors of this paper in an earlier publication [Fischler83]. The algorithm was effective in finding such structure, but it provided no mechanism for distinguishing between the semantically meaningful objects and the "accidental" and irrelevant linear features found in most real images. In work now in progress, we use the SSS algorithm to "slice up" the individual curves found by the delineation algorithm. We throw away the very small resulting segments which are typical of accidental linear formations, and then further filter the longer segments with respect to a set of semantic constraints. Those segments that pass through the filtering process are then "glued" back together to produce the desired delineation. This process is illustrated in Figure 6. Figure 6a shows an aerial image, and 6b shows the linear segments extracted by use of the original delineation algorithm. Figure 6c shows those segments that passed through the filters mentioned above, and Figure 6d shows the result of a final step to retain only the more significant roads and trails. The two panes of Figure 6e show the results of applying the SSS algorithm to some of the 120 curves highlighted in Figure 6b (they have been isolated and separated into the two panes to allow clear display of the partition points and

to prevent confusion due to the intersections of distinct curves). The robustness of the SSS algorithm is essential in carrying out the filtering operation. Insertion of extraneous partition points would cause the loss of portions of the road network; absence of valid partition points would allow meaningless appendages to become part of the extracted network.

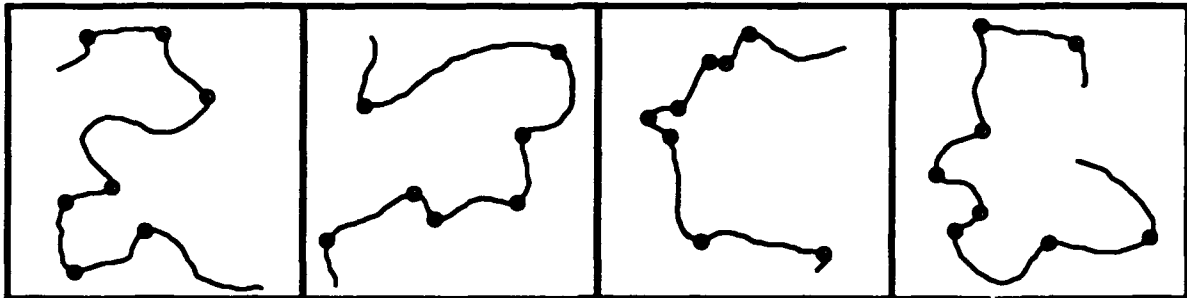
10. References

1. F. Attneave, "Some informational aspects of visual perception," *Psychol. Rev.* 61:183-193, 1954.
2. A. Bengtsson and J.O. Eklundh, "Shape Representation by Multiscale Contour Approximation," *IEEE Trans PAMI-13*(1):85-93, Jan. 1991.
3. A.K. Cline, "Scalar- and planar- valued curve fitting using splines under tension," *CACM* 17(4):218-223, April 1974.
4. L.S. Davis, "Understanding shape: angles and sides," *IEEE Trans. Comput.* C-26:236-242, March 1977.
5. M.A. Fischler and R.C. Bolles, "Perceptual organization and curve partitioning," *IEEE Trans PAMI-8*(1):100-105, Jan. 1986.
6. M.A. Fischler and H.C. Wolf, "Linear Delineation," *Proceedings IEEE CVPR-83*, June 1983, pp 351-356; also, *Readings in Computer Vision* (M.A. Fischler and O. Firschein, eds.), Morgan Kaufmann, pp 204-209, 1987.
7. M.A. Fischler and P. Barrett, "An iconic transform for sketch completion and shape abstraction," *CGIP* 13:334-360, 1980.
8. D. Hilbert and S. Cohen-Vossen, "Geometry and the imagination." Chelsea, 1952.
9. D.D. Hoffman and W.A. Richards, "Representing smooth plane curves for recognition: implications for figure-ground reversal," *Proc. 2nd Nat. Conf. Artificial Intelligence*, Pittsburgh, PA, pp 5-8, Aug. 1982.
10. H. Imai and M. Iri, "Computational-geometric methods for polygonal approximations of a curve," *CVGIP-36*(1):31-34, Oct. 1986.
11. D.G. Lowe, "Organization of smooth image curves at multiple scales," *Proc 2nd ICCV*, pp. 558-567, 1988.
12. R. Mehrotra, S. Nichani, and N. Ranganathan, "Corner detection," *Pattern Recognition* 23(11):1223-1233, 1990.
13. F. Mokhtarian and A. Mackworth, "Scale-based description and recognition of planar curves and two-dimensional shapes," *IEEE PAMI* 8(1):34-43, Jan 1986.
14. T. Pavlidis and S.L. Horowitz, "Segmentation of plane curves," *IEEE Trans. Comput.* C-23:860-870, Aug. 1974.
15. W. Richards and D. Hoffman, "Codon constraints on closed 2D shapes," in *Human and Machine Vision II* (A. Rosenfeld, ed.), Academic Press, pp 207-223, 1986.
16. W. Richards, B. Dawson, and D. Whittington, "J. Optical Soc. Amer." 3(9):1483-1491, Sept. 1986.
17. A. Rosenfeld and E. Johnston, "Angle detection in digital curves," *IEEE Trans. Comput.* C-22:875-878, 1973.
18. A. Rosenfeld and J.S. Weszka, "An improved method of angle detection on digital curves." *IEEE Trans. Comput.* C-24:940-941, Sept. 1975.
19. C.H. Teh and R.T. Chin, "On the detection of dominant points on digital curves," *IEEE Trans PAMI-11*(8):859-872, Aug. 1989.
20. A. Witkin, "Scale Space Filtering," *Proc. 8th IJCAI*, Karlsruhe, West Germany, pp 1019-1022, Aug. 1983.
21. D.M. Wuescher and K.L. Boyer, "Robust contour decomposition using a constant curvature criterion," *IEEE Trans PAMI-13*(1):41-51, Jan. 1991.

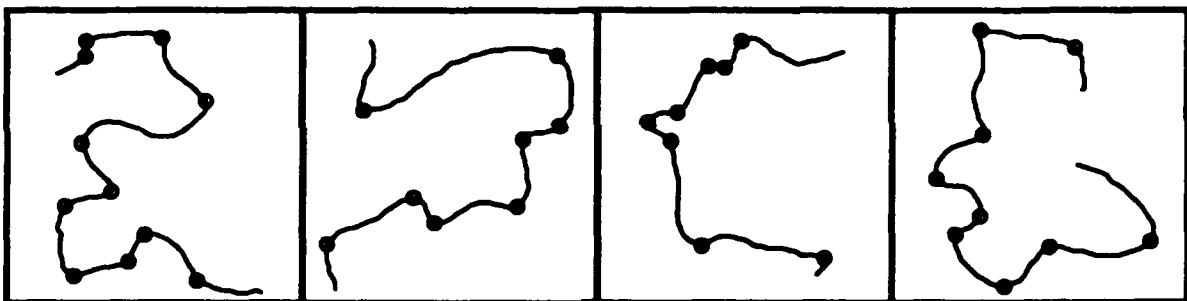
CURVE PARTITIONING: Instructions

For each enclosed curve:

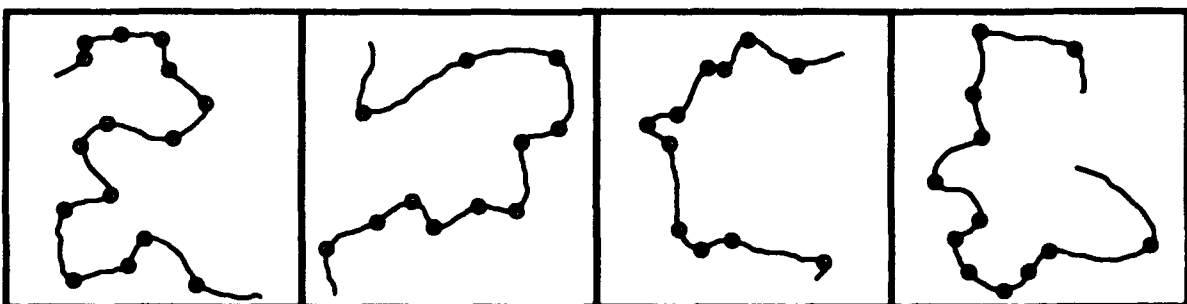
Assume that 10 years from now you will be asked to reconstruct the given curve. A reasonably correct reconstruction will be rewarded by a large sum of money (say \$5000). You can record, for later use, the locations of up to nine points along the curve to help you do the reconstruction – but it will cost you \$200 for each such point (to be subtracted from your prize if you receive the reward). Please mark your selected points on the curve. Do not select the endpoints, they will be provided free. Do not take more than one minute per curve.



Points chosen by 9 of 11 test subjects

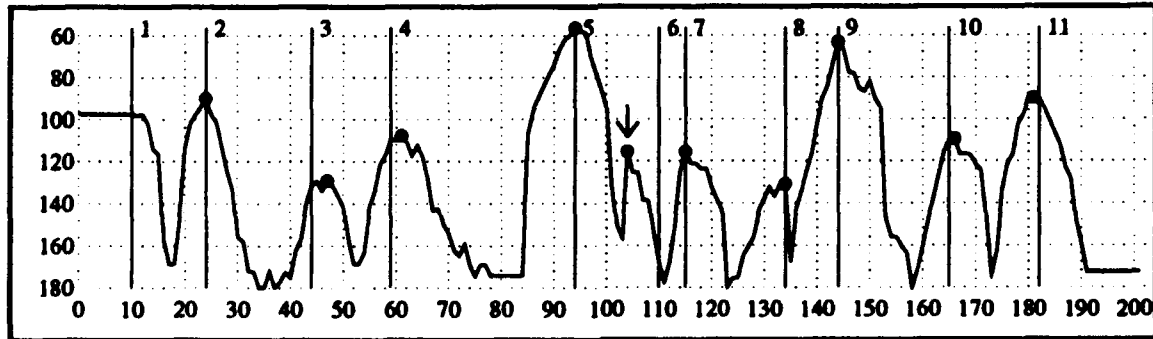
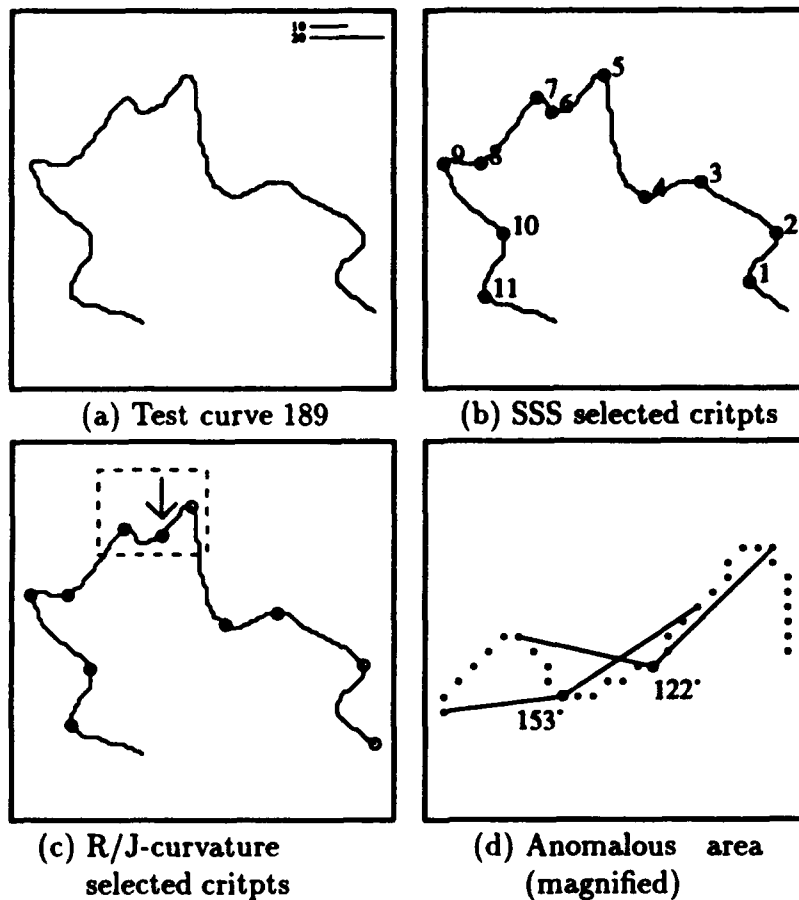


Critical points found by the SSS algorithm



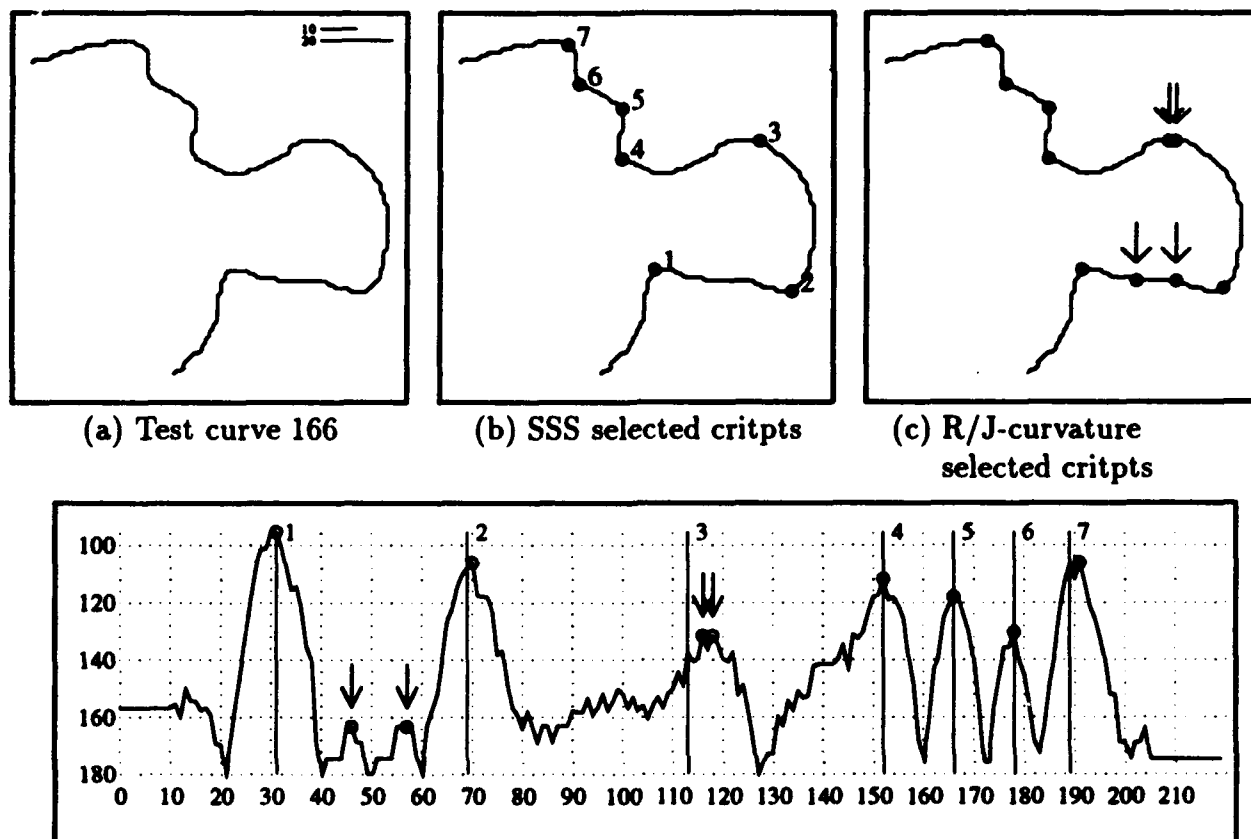
Points chosen by at least 1 of 11 test subjects

Figure 1: Comparison of human and SSS algorithm performance in the curve partitioning task. (Each of the curves used in the experiments with human subjects was contained in a square that was 1.5 inches on a side.)



(e) Plot of R/J-curvature along test curve. Abcissa = sequence number of point on curve. Ordinate = angle (in degrees) computed at point. (Angle-arms are 10 units each for R/J-C; standard stick lengths of 10 and 20 units are employed by SSS.)

Figure 2: Comparison of SSS and R/J-curvature metrics evaluated on test curve 189. The continuous curve in (e) represents R/J-curvature along the test curve shown in (a). The vertical lines in (e) mark the sequentially numbered critpts selected by SSS as shown in (b). The critpts corresponding to the extreme values of R/J-curvature shown in (c) are marked as circles in (e). The arrow in (c), and in the corresponding location in (e), illustrates an anomalous selection using R/J-curvature. (d) shows the computed values of R/J-curvature, 153°, at the preferred location and 122° at the location of the anomalous selection.



(d) Plot of R/J-curvature along test curve. Abcissa = sequence number of point on curve. Ordinate = angle (in degrees) computed at point. (Angle-arms are 10 units each for R/J-C; stick length is 20 units for F/B-S.)

Figure 3: Comparison of SSS and R/J-curvature metrics evaluated on test curve 166. The continuous curve in (d) represents R/J-curvature along the test curve shown in (a). The vertical lines in (d) mark the sequentially numbered critpts selected by SSS as shown in (b). The critpts corresponding to the extreme values of R/J-curvature shown in (c) are marked as circles in (d). The arrows in (c), and in the corresponding locations in (d), illustrates anomalous selections using R/J-curvature.

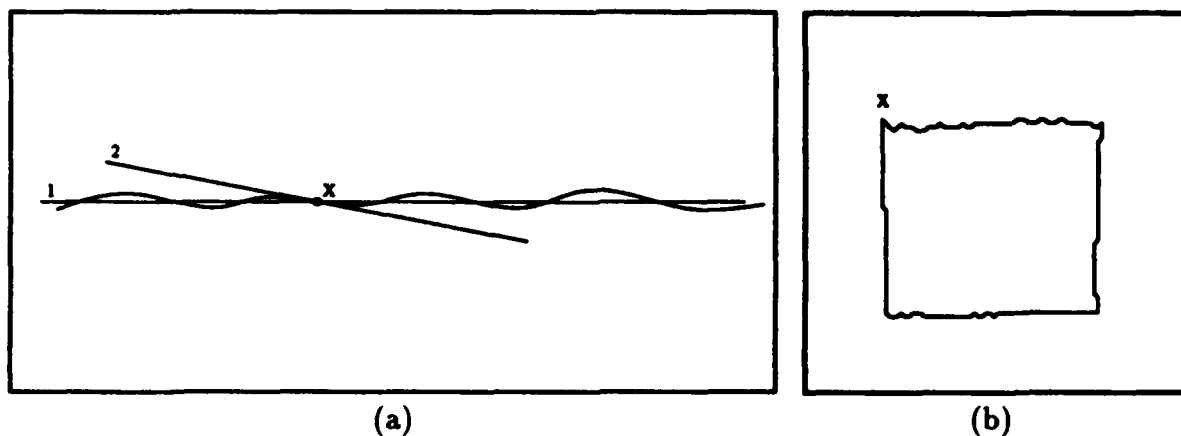


Figure 4: Curvature and saliency are functions of curve resolution. As illustrated in (a) above, we can draw more than one visually acceptable tangent to many of the points on this curve at the given resolution. As resolution increases, tangent 2 would dominate at point x ; as resolution decreases, tangent 1 would dominate at the same point. In (b), the angle at x can be seen as 45° at one scale and 90° at a larger scale. Thus, curvature and saliency are not unique properties of curve points.

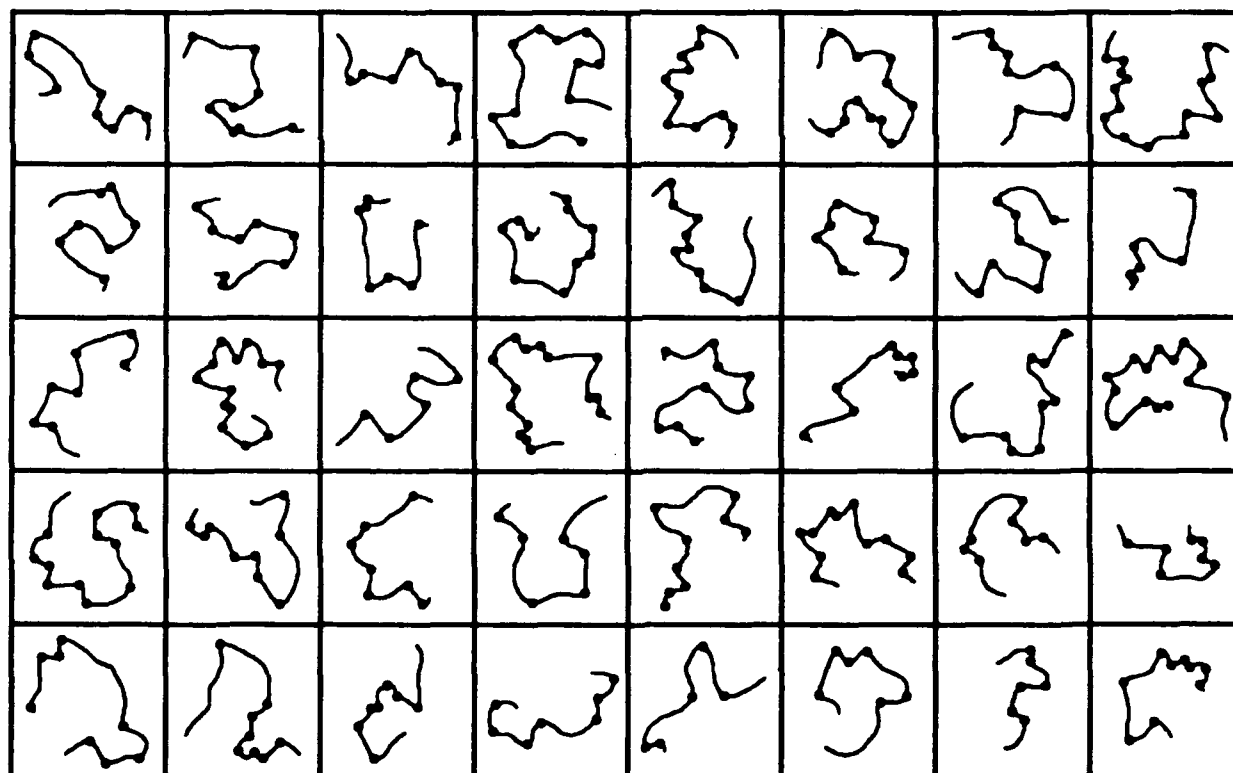
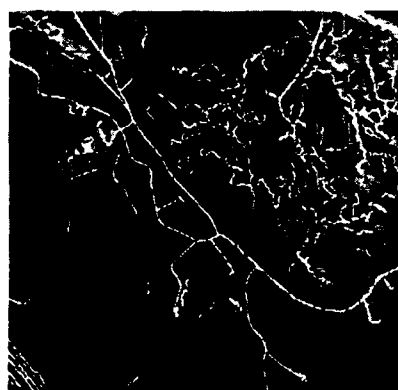


Figure 5: Critical points found by the SSS algorithm for a set of 40 random curves.



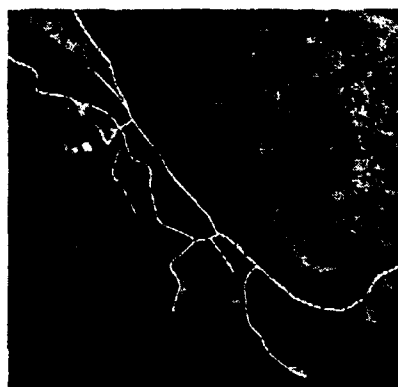
(a) Aerial photograph



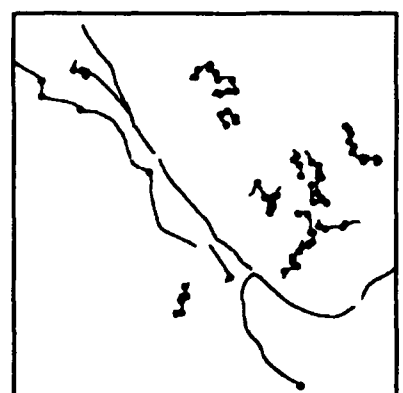
(b) Initial extraction of linear structure



(c) Filtered linear structure using SSS algorithm



(d) Delineation of major roads and trails



(e) Partition points found by SSS algorithm on curves from (b)

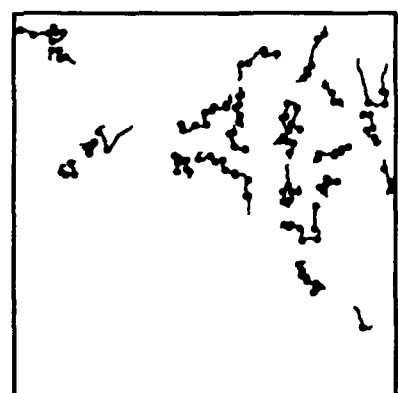


Figure 6: Application of the SSS algorithm to the problem of delineating linear features in aerial photographs.

Detecting Occluding Edges Without Computing Dense Correspondence

Lambert E. Wixson*
Computer Science Dept.
University of Rochester
Rochester, NY 14627
email: wixson@cs.rochester.edu

Abstract

This paper presents an algorithm for detecting occluding edges in stereo pairs. The algorithm described here does not require dense correspondence estimates and hence may be applied in a selective manner. It extracts intensity edges and tests them for occlusion by sampling edgels along each edge. The algorithm works by searching for matches, in the right image, for the regions to the left and right of a left-image edgel. The method is based on that of Toh and Forrest [1990], but extends that work in several ways. First, it adds an algorithm for automatically selecting an appropriate size for the correlation windows used to detect the occlusions. Second, it adds a simple technique for classifying an entire edge by sampling only a few pixels along the edge. Finally, it identifies two situations that lead to false positive and false negative classifications, and describes solutions to these problems.

1 Introduction

Occluding edges provide important information about one's environment. Knowledge of the locations of occluding edges can be used to increase the robustness of object recognition programs [Thompson and Whillock, 1988]. Also, occluding edges can denote surface discontinuities such as holes or cliffs. Finally, occluding edges identify the borders of regions that cannot be imaged from the current camera viewpoint, and hence can be used to guide a camera to look into these regions. For all of these uses, it seems sufficient to view occluding edges as qualitative phenomena; it does not seem necessary to construct a dense depth map.

Traditionally, however, occlusions have been detected as a side-effect of the construction of dense correspondence maps. The most widely-described

occlusion-detection techniques attempt to detect not occluding edges, but rather occluded *points* in a stereo or motion pair [Mutch and Thompson, 1985, Geiger *et al.*, 1992, Jones and Malik, 1992, Weng *et al.*, 1992]. These are points that appear in one image but do not appear in the other, and hence cannot be matched from one image to the other. Such points appear in the vicinity of occluding edges. Because of the difficulties of determining matching points, occluded-point detection is typically incorporated into algorithms for determining dense correspondence maps. Unfortunately, such algorithms are time-consuming, and even once the occluded points have been located, the occluding edges still must be inferred. Methods that do not involve dense correspondence and that directly identify occluding edges would be more desirable.

Three such methods have appeared in the literature. The first uses active adjustment of a camera's depth-of-field [Toh and Forrest, 1990, Brunnstrom *et al.*, 1991]. The remaining two techniques involve stereo pairs. Little [1990] identifies points near an occlusion edge by examining the distribution of match goodnesses as a window around the point is shifted horizontally. If the window overlaps surfaces at different depths, this distribution is bimodal. Finally, [Toh and Forrest, 1990] presents a method that examines two adjacent image patches. Its operation is illustrated in Figure 1. Given a near-vertical edgel¹ e in the left image L , this algorithm operates by selecting two windows in L . Window P_L runs leftwards from e , while window N_L runs rightwards from e . The right image R is then searched, along roughly the epipolar line, for the regions P_R and N_R that best match P_L and N_L , respectively. Let C_P and C_N be

¹An edgel is simply a pixel at which a significant intensity gradient exists. This paper describes only the use of near-vertical edgels because it is effectively impossible to match horizontal edgels in left and right stereo pairs — usually a match at one disparity is indistinguishable from a match at another. Of course, the same algorithms could be applied to near-horizontal edgels given a top and bottom stereo pair.

*This material is based on work supported by DARPA Contract MDA972-92-J-1012.

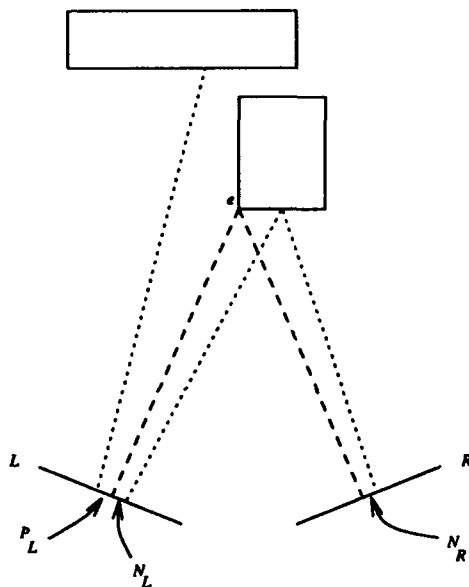


Figure 1: Visual cue indicating the presence of an occluding edge. If e is a right-occluding edge, then the surface to its right spanned by N_L will be visible in the right image. On the other hand, the surface to e 's left is blocked from the right image's view. As a result, the region spanned by P_L is not visible in the right image.

the goodness-of-match between P_L and P_R , and N_L and N_R , respectively. If C_P and C_N are both high, and P_L and P_R are next to each other in R , this means that the regions on both sides of e were visible in the right image, and therefore that e is an intensity edge but not an occluding edge. If C_P (C_N) is small, however, then this means that P_L (N_L) is not visible in the right image, and that e is a right-(left)-occluding edge. (A right-(left)-occluding edge is an edge such that the surface to its right (left) is closer to the camera.)

This paper describes experience with Toh and Forrest's algorithm. It contains solutions to two problems that arise in practice, namely how to select P_L and N_L so that they contain enough texture for the matching algorithm to be able to truly determine whether or not matches exist in the right image, and how to use the detector to rapidly classify edges. The paper also identifies two situations that can cause misclassifications. First, if diagnosis is based only on C_P and C_N , situations exist in which false negative responses can be obtained, i.e. occluding edgels can be classified as a surface marking. Second, false positive responses can also be obtained — a non-occluding edge e will be classified as an occluding edge if a true occluding edge is closer to the camera and passes through e 's P_L or N_L window. Solutions to these problems are pre-

sented. Additional details can be found in [Wixson, 1993].

2 Selecting the width of the matching windows

Consider the stereo image pair shown in Figure 4. It contains little surface texture near many of its intensity edges. Yet many occluding edges can be seen in this image pair by comparing the distance between pairs of edges. For example, consider the distance between the left edge of the detergent box and the right edge of the cylinder to its left. In the left image, this distance is larger than in the right image, and therefore it follows that the left edge of the box is a right-occluding edge, i.e. an occluding edge whose right side is closer to the camera than its left. Similar inferences can be made based on the relationship between the right side of the cardboard box and the left edge of the decahedron and the coffee cup and the decahedron. At the same time, of course, the distances between many of the surface markings on the detergent box are the same in both images and hence these are not occluding edges. Thus we see that in order to detect occluding edges in untextured scenes, the match windows must be wide enough to reach the next adjacent edge.

This suggests that a good strategy for selecting the width of the matching window might simply be to make the window wide enough to contain the closest strong intensity edge. This can be done as follows: To find the right bound x_N of the N_L window to the right of edgel x_e, y_e , step rightwards from x_e until the x -derivative exceeds a threshold δ and from there keep stepping rightwards until the x -derivative falls below a threshold ϵ . The purpose of stepping until falling below ϵ is to ensure that enough of the adjacent edge is contained within the window for it to influence the goodness of match.

To ensure that the window is big enough to allow robust matching, a minimum distance m between x_e and x_N is required. If $x_N - x_e < m$, x_N is reset to $x_e + m$. On the other hand, no limit is imposed on the maximum window width.

The left bound x_P of the left match window P_L is computed similarly. If the algorithm for selecting x_P or x_N runs off the side of the image before a pixel is found that meets the criteria, the edgel x_e, y_e is classified as "unmeasurable".

To increase robustness, both match windows are given some vertical extent, i.e. are more than one row high. However, this amount is fixed rather than adaptive.

The above method for selecting window size is *ad hoc* but effective. A more principled method is described

3 Matching

Because the sizes of the correlation windows vary depending upon the surroundings of the edgel being tested, and we would like to have a single threshold that determines whether a match is a good match, the similarity measure must be independent of window size. Therefore, match goodnesses between windows in the left and right image are computed using normalized correlation [Ballard and Brown, 1982, p. 68]:

$$C(p, q) = \frac{\overline{pq} - \bar{p} \bar{q}}{\sigma(p) \sigma(q)} \quad (1)$$

where p and q are image patches, and $\sigma(p)$ and $\sigma(q)$ are the standard deviations of the pixel values in the patches.

4 Analyzing the matching results

Given the windows P_R and N_R in the right image that best match P_L and N_L , let C_P and C_N be the corresponding similarity measurements between the windows, and let d_{PN} be the distance, in pixels, between the right edge of P_R and the left edge of N_R . Let τ be the threshold over which a match is considered to be a good match. Then the diagnosis as to the nature of the edgel e that P_L and N_L abut is determined according to Table 1. An edgel can be diagnosed either as a surface marking, a left-occluding edgel (which means that the left side of the edgel is the closer surface), a right-occluding edgel, or as a point that is not visible in the other image.

Table 1 is for the most part straightforward, except for the case in its upper-left corner, where both C_P and C_N indicate good matches. In this case, it is necessary to examine d_{PN} , because there are two reasons why there could be good matches for both P_L and N_L . The first, of course, is that e might only be a surface marking, not an occluding edge. As a result, d_{PN} should be small. The second possibility is that e is a left-occluding edge, as illustrated in Figure 2. In this case, the regions imaged by the P_L and N_L windows in the left image are both imaged in the right image, but the right camera will also image some more of the background surface between the P_R and N_R windows. An example of this phenomenon occurs with the right side of the detergent box, a left-occluding edge, in Figure 4. In the left image, much less of the cardboard box behind the detergent box is visible. The portion of the cardboard box visible in the left image is also

		$C_N \geq \tau$					
		yes	no				
$C_P \geq \tau$	yes	$d_{PN} \leq \Delta$ <table> <tr> <td>yes</td> <td>no</td> </tr> <tr> <td>marking</td> <td>left-occ</td> </tr> </table>	yes	no	marking	left-occ	left-occ
	yes	no					
marking	left-occ						
no		right-occ	not-visible				

Table 1: Diagnosis procedure for determining the nature of an edgel given C_P , C_N , and d_{PN} . C_P and C_N are goodnesses-of-match between P_L and P_R , and N_L and N_R , respectively, where P_R and N_R are the windows in the right image that best match P_L and N_L . d_{PN} is the distance in pixels between the right edge of P_R and the left edge of N_R .

visible in the right image, but the right image also contains a region of the cardboard box to the left of this portion.

5 Classifying edges based on sparse measurements

One advantage of the occlusion detection method presented above is that, since it is non-iterative and does not involve smoothing, it does not require dense measurement of the pixel disparities. As a result, the algorithm can be selectively applied only to points of interest, saving computation time. One obvious method for selecting these points of interest is to select them along previously-extracted intensity edges. This approach has several advantages. If long edges are more likely to be important than short edges, the edges can be tested in order of decreasing length, thus creating a type of anytime algorithm. Also, very short edges can be discarded to avoid wasting effort testing them for occlusion.

Using an intensity edge to select edgels to be tested for occlusion has one additional advantage. Edgels that are on the same edge are likely to have the same properties. If, for instance, several edgels along the same intensity edge are left-occluding edgels, then the other edgels along the edge are highly likely to be left-occluding. This suggests that an intensity edge can be efficiently tested for occlusion by testing only a subset of its edgels. In our implementation, we test only every k 'th (typically $k = 7$) edgel along an edge. Of these, we discard all edgels discovered to be unmeasurable. If more than 20% of the remaining tested edgels are determined to be right-

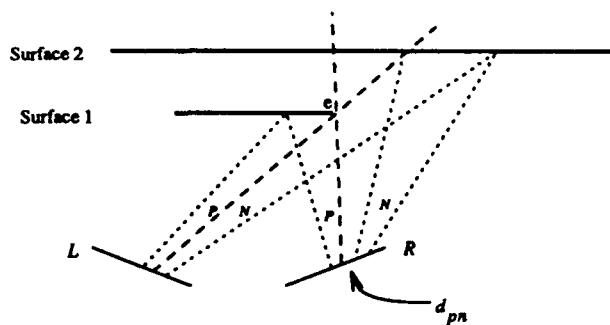


Figure 2: A left-occluding edge (e) may not be detectable using only the match goodnesses for the left image's P and N windows. Here the regions imaged in the left P and N are also viewable in the right image. The clue that e is a left-occluding edge is that there is a gap between the right image's match to P and its match to N . The width of this gap is denoted by d_{PN} .

(left-)occluding, then the edge is classified as a right-(left-)occluding edge. If both the number of left-occluding edgels and the number of right-occluding edgels exceed 20%, the edge is classified according to whichever has the most votes.

6 Experimental results

The above algorithm was applied to the stereo pairs in Figures 4, 5, and 6. Vertical edges were extracted by chaining together pixels at which the absolute value of the intensity x -derivative exceeded δ and was a local maxima. Chains of length < 8 were discarded. The remaining edges are shown in the figures' third column.

The edges were classified using the incremental test strategy described in the previous section. This strategy resulted in a significant reduction of computation; for each of the 240×244 pairs in Figures 4-6, the number of edgels tested for occlusion was between 300 and 500, which is much less than the total number of either pixels or edgels in the images. The parameter values used to determine the width of the match windows were $\delta = 6$, $\epsilon = 1.1$, and $m = 13$. Using these parameters on images that were 244 columns wide, the median width of the resulting windows selected in these images was between 15 and 20 pixels, while the average width was 25 pixels. The match windows were $2v + 1$ pixels high, and v was 3.

The parameter values used to diagnose edgels were $\tau = .75$, and $\Delta = 3$. The m , τ , and Δ parameters are quite robust and are unlikely to require modification from image to image. The δ and ϵ parameters, which are thresholds based on measured derivative

magnitudes, may require modification depending on the sharpness of the image. These were selected by examining the measured derivatives.

The identified occluding edges are displayed in the rightmost column of Figures 4-6. Right-occluding edges are displayed as solid lines, while left-occluding edges are dotted lines. The figures show that most surface markings and edges that are too distant to have significant stereo disparity are filtered out by the occlusion detector, and that it succeeds in detecting many of the true occluding edges, such as the right-occluding edges created by the left edges of Figure 4's detergent box, mug, and decahedron, and the left edges of the chair seat and chair back in Figure 5.

Note that in some cases, significant vertical occluding edges were not labeled as occluding edges. Notably missing in Figure 4 are the top section of the detergent box's left edge, the detergent box's right edge, both edges of the cylinder, the left edge of the tape dispenser, and the right edge of the decahedron. These edges were unmeasurable because on one side of them, no matching window could be selected; the border of the image was encountered before another significant edge could be found that could serve as the match window border.

7 False positives

Inspection of the results in Figures 4-6 reveals some falsely detected occlusions. There are three causes of these false positives. The first is a problem with all correlation-based matching algorithms — they fail to obtain good matches on curved surfaces or surfaces that slope sharply away from the stereo baseline. Therefore, a surface marking on the mug in Figure 4 and the nearest edge of the chair back in Figure 5 are falsely classified as occluding edges, and the right side of the white box in Figure 6 is misclassified as a left-occluding edge.

The second, relatively minor, cause of some false positives is failure to detect the proper terminating edge of the P_L or N_L window. This occurred for the lower right quadrant of the circle on the detergent box in Figure 4. The portion of the detergent box's right edge that is to the right of these edgels does not give rise to a significant image gradient. As a result, their N_L windows are extended to the right edge of the box behind the detergent box. Since this distance is not the same in the right image, the N_L match fails and the edgels are diagnosed as left-occluding.

The third cause of false positives is fairly important and is most clearly illustrated by the wall socket that appears near the chair back in Figure 5. Its left edge

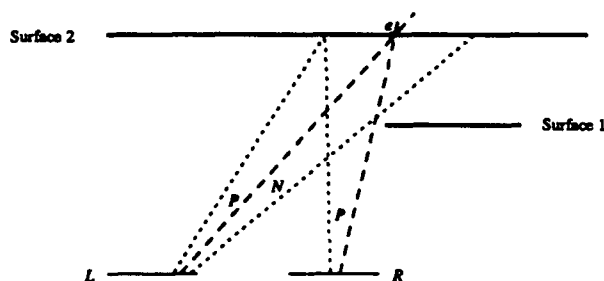


Figure 3: A occluding edge in the foreground may cause a surface marking to be incorrectly diagnosed as an occluding edge. Here, for surface marking e , the area imaged in the left image's N window is not visible in the right image due to a closer surface. As a result, C_N is small and the classification procedure described in Table 1 will diagnose e as a left-occluding edge.

is not an occluding edge, yet is detected as such. As illustrated in Figure 3, this occurs because no match can be found for the N_L windows emanating from edgels on the left edge of the socket. No match can be found because that region in the right image is occluded by the chair back. Thus, surface markings that project to image locations near the projections of closer occluding edges may be themselves incorrectly diagnosed as occluding edges. Other examples of this appear to the left of the left edge of the chair seat in Figure 5. A heuristic for eliminating these false positives via a postprocessing step is described in [Wixson, 1993].

8 Conclusion

This paper has described an occlusion detection algorithm that does not require dense correspondence estimates and hence can be applied selectively. Currently, its output is rather qualitative; it classifies intensity edges as either left- or right-occluding, but does not estimate the size of the depth discontinuity across the edge, which would provide a hint as to the size of the occluded region. However, the algorithm could be augmented to estimate the depth discontinuity magnitude by computing the relative disparity of non-occluded regions near each side of an occluding edge.

Acknowledgments

I thank Chris Brown, Polly Pook, and Ray Rimey for their helpful comments.

References

- [Ballard and Brown, 1982] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice Hall, Inc., 1982.
- [Brunnstrom et al., 1991] Kjell Brunnstrom, Tony Lindeberg, and Jan-Olof Eklundh. Active detection and classification of junctions by foveation with a head-eye system guided by the scale-space primal sketch. CVAP Report 97, Royal Institute of Technology, Stockholm, Sweden, 1991.
- [Geiger et al., 1992] Davi Geiger, Bruce Ladendorf, and Alan Yuille. Occlusions and binocular stereo. In *Proceedings of the Second European Conference on Computer Vision*, 1992.
- [Jones and Malik, 1992] David G. Jones and Jitendra Malik. A computational framework for determining stereo correspondence from a set of linear spatial filters. In *Proceedings of the Second European Conference on Computer Vision*, 1992.
- [Kanade and Okutomi, 1990] Takeo Kanade and Masatoshi Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. In *Proceedings of the DARPA Image Understanding Workshop*, September 1990.
- [Little and Gillett, 1990] James J. Little and Walter E. Gillett. Direct evidence for occlusion in stereo and motion. In *Proceedings of the European Conference on Computer Vision*, 1990.
- [Mutch and Thompson, 1985] Kathleen M. Mutch and William B. Thompson. Analysis of accretion and deletion at boundaries in dynamic scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 133–138, March 1985.
- [Thompson and Whillock, 1988] W.B. Thompson and R.P. Whillock. Occlusion-sensitive matching. In *Proceedings of the International Conference on Computer Vision*, 1988.
- [Toh and Forrest, 1990] Peng-Seng Toh and Andrew K. Forrest. Occlusion detection in early vision. In *Proceedings of the International Conference on Computer Vision*, 1990.
- [Weng et al., 1992] Juyang Weng, Narendra Ahuja, and Thomas S. Huang. Matching two perspective views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 806–825, August 1992.
- [Wixson, 1993] Lambert E. Wixson. Detecting occluding edges by examining adjacent image patches. Submitted to IEEE Workshop on Qualitative Vision, 1993.

[Ballard and Brown, 1982] D.H. Ballard and C.M.

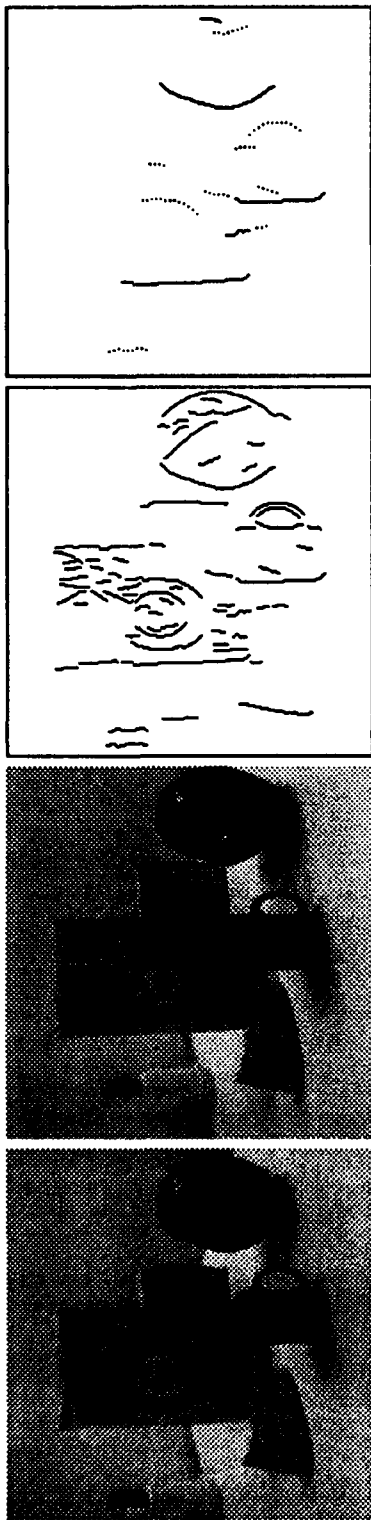


Figure 4: "Detergent" stereo pair, extracted edges, and detected occluding edges.

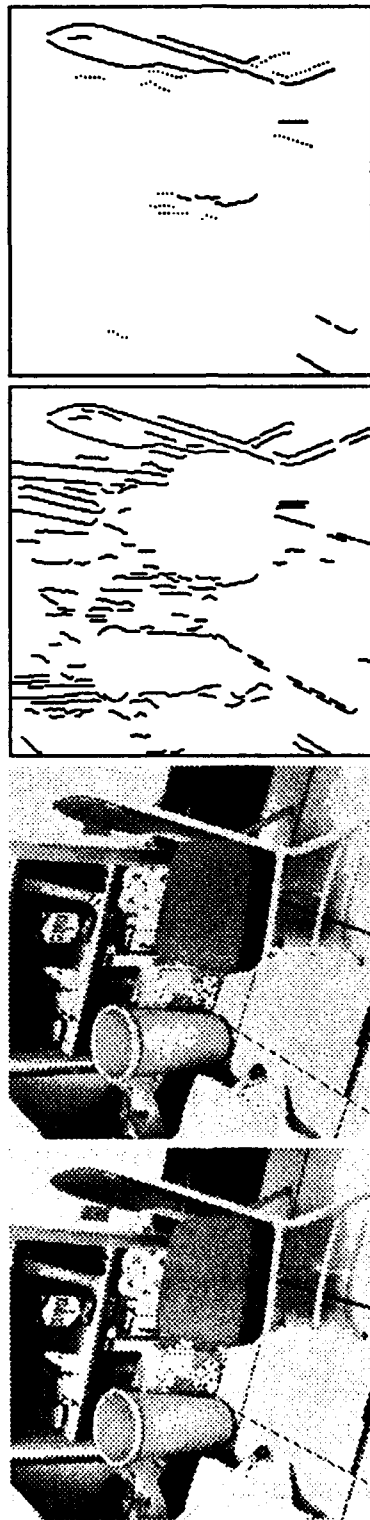


Figure 5: "Chair" stereo pair, extracted edges, and detected occluding edges.



Figure 6: "Desk" stereo pair, extracted edges, and detected occluding edges.

Robust Shape Recovery from Occluding Contours Using a Linear Smoother

Richard Szeliski

Digital Equipment Corporation,
Cambridge Research Lab,
One Kendall Square, Bldg. 700,
Cambridge, MA 02139

Richard Weiss

Computer and Information Science Department,
University of Massachusetts at Amherst,
Amherst, MA 01003

Abstract

Recovering the shape of an object from two views, or stereo, fails at occluding contours of smooth objects because the extremal contours are view dependent. For three or more views, shape recovery is possible, and several algorithms have recently been developed for this purpose. We present a new approach to the multiframe shape recovery problem which does not depend on differential measurements in the image, which may be noise sensitive. Instead, we use a linear smoother to optimally combine all of the measurements available at the contours (and other edges) in all of the images. This allows us to extract a robust and dense estimate of surface shape, and to integrate shape information from both surface markings and occluding contours.

1 Introduction

Most visually-guided systems require representations of surfaces in the environment in order to integrate sensing, planning, and action. The task considered in this paper is the recovery of 3D structure (shape) of objects with piecewise-smooth surfaces from a sequence of profiles taken with known camera motion. The *profile* (also known as the *extremal boundary* or *occluding contour*) is defined as the image of the *critical set* of the projection map from the surface to the image plane. Since profiles are general curves in the plane without distinguished points, there is no *a priori* pointwise correspondence between these curves in different views. However, given the motion, there is a correspondence based on the *epipolar constraint*. For two images, i.e., classical *stereo*, this epipolar constraint is a set of straight lines. These lines are the intersection of the *epipolar planes* with the image plane. The epipolar plane through a point is determined by the view direction at that point and the camera translation direction.

In the case of contours that are not view dependent, e.g., creases (tangent discontinuities) and surface markings, many techniques have been developed

for recovering the 3D contour locations from two or more images under known camera motion [Marr and Poggio, 1979; Mayhew and Frisby, 1980; Arnold, 1983; Bolles *et al.*, 1987; Baker and Bolles, 1989; Matthies *et al.*, 1989]. However, for smooth curved surfaces the critical set which generates the profile is different for each view. Thus, the triangulation applied in two-frame stereo will not be correct along the occluding contour for smooth surfaces. For the same reason, it is not possible to determine the camera motion from the images unless some assumptions are made either about the surface or the motion [Arbogast and Mohr, 1992; Giblin *et al.*, 1992]. On the other hand, the fact that the critical sets sweep out an area means that the connectivity of the surface points can be determined, i.e. one obtains a surface patch rather than a set of points.

The problem of reconstructing a smooth surface from its profiles has been explored for known planar motion by Giblin and Weiss [1987] and subsequently for more general known motion by Vaillant [Vaillant, 1990; Vaillant and Faugeras, 1992] and Cipolla and Blake [Blake and Cipolla, 1990; Cipolla and Blake, 1990; Cipolla and Blake, 1992]. These approaches are based on a differential formulation and analysis or use three frames. Unfortunately, determining differential quantities reliably in real images is difficult. This has led Cipolla and Blake to use relative measurements in order to cancel some of the error due to inadvertent camera rotation. Their approach used B-snakes which require initialization for each contour that is tracked. In addition, B-snakes implicitly smooth the contours in the image. Since the recovery of 3D points is a linear problem, the smoothing can be done in 3D on the surface where more context can be used in the detection of discontinuities, so that detailed structure can be preserved.

To overcome these limitations, the approach we develop in this paper applies estimation theory (Kalman filtering and smoothing) to make optimal use of each measurement without computing differential quantities. First, we derive a *linear* set of equations between the unknown shape (surface point positions and radii of curvature) and the measurements. We then develop a robust linear smoother ([Gelb, 1974; Bierman, 1977]) to compute statistically optimal current and past estimates from the set of contours. Smoothing allows us to combine measurements on both sides of each surface point.

*The second author acknowledges the support of grants TACOM DAAE07-91-C-RO35, NSF IRI-9208920, and NSF IRI-9116297.

Our technique produces a complete surface description, i.e., a network of linked 3D surface points, which provides us with a much richer description than just a set of 3D curves. Due to self-occlusion and occlusion by other surfaces, some parts of the surface may never appear on the profile. Since the method presented here also works for arbitrary surface markings and creases, a larger part of the surface can be reconstructed than from occluding contours of the smooth pieces alone. Our approach also addresses the difficult problem of contours that merge and split in the image, which must be resolved if an accurate and complete 3D surface model is to be constructed.

The method we develop has applications in many areas of computer vision, computer aided design, and visual communications. The most traditional application of visually based shape recovery is in the reconstruction of a mobile robot's environment, which allows it to perform obstacle avoidance and planning tasks [Curwen *et al.*, 1992].

Our paper is structured as follows. We begin in Section 2 with a description of our edge detection, contour linking, and edge tracking algorithms. In Section 3, we discuss the estimation of the epipolar plane for a sequence of three or more views. Section 4 presents the linear measurement equations which relate the edge positions in each image to the parameters of the circular arc being fitted at each surface point. Section 5 then reviews robust least squares techniques for recovering the shape parameters and discusses their statistical interpretation. Section 6 shows how to extend least squares to a time-evolving system using the Kalman filter, and develops the requisite forward mapping (surface point evolution) equations. Section 7 extends the Kalman filter to the linear smoother, which optimally refines and updates previous surface point estimates from new measurements. Section 8 presents a series of experiments performed both on noisy synthetic contour sequences and on real video images. We close with a discussion of the performance of our new technique and a discussion of future work.

2 Contour detection and tracking

The problem of edge detection has been extensively studied in computer vision [Marr and Hildreth, 1980; Canny, 1986]. The choice of edge detector is not crucial in our application, since we are interested mostly in detecting strong edges such as occluding contours and visible surface markings.¹ For our system, we have chosen the *steerable filters* developed by Freeman and Adelson [1991], since they provide good angular resolution at moderate computation cost, and since they can find both step and peak edges. An example of our edge detector operating on the input image in Figure 1a is shown in Figure 1b.

Once discrete edgels have been detected, we use local search to link the edgels into contours. We find the two

¹Unlike many edge detection applications, however, our systems does provide us with a quantitative way to measure the performance of an edge detector, since we can in many cases measure the accuracy of our final 3D reconstruction.

neighbors of each edgel based on proximity and continuity of orientation. Note that in contrast to some of the previous work in reconstruction from occluding contours [Cipolla and Blake, 1990; Cipolla and Blake, 1992], we do not fit a smooth parametric curve to the contour since we wish to directly use all of the edgels in the shape reconstruction, without losing detail.

We then use the known epipolar lines (Section 3) to find the best matching edgel in the next frame. Our technique compares all candidate edgels within the epipolar line search range (defined by the expected minimum and maximum depths), and selects the one which matches most closely in orientation (see Figure 1c).

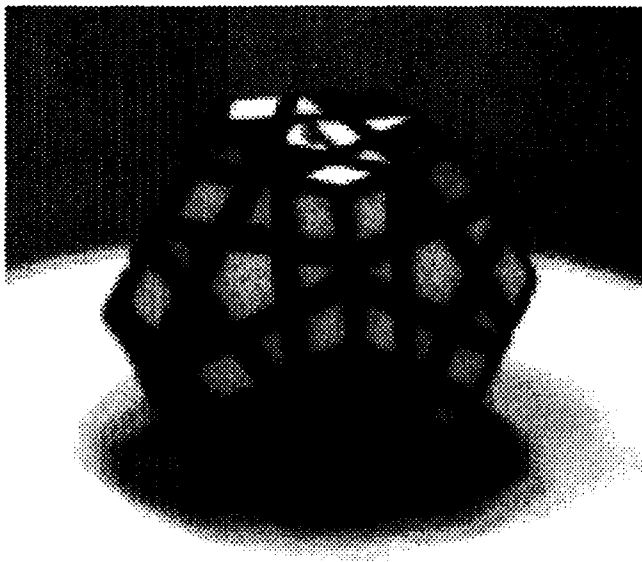
Since contours are maintained as a list of discrete points, it is necessary to resample the edge points in order to enforce the epipolar constraint on each track. We occasionally start new tracks if there is a sufficiently large (2 pixel wide) gap between successive samples on the contour. While we do not operate directly on the spatiotemporal volume, our tracking and contour linking processes form a virtual surface similar to the *weaving wall* technique of Harlyn Baker [1989]. Unlike Baker's technique, however, we do not assume a regular and dense sampling in time.

3 Reconstructing surface patches

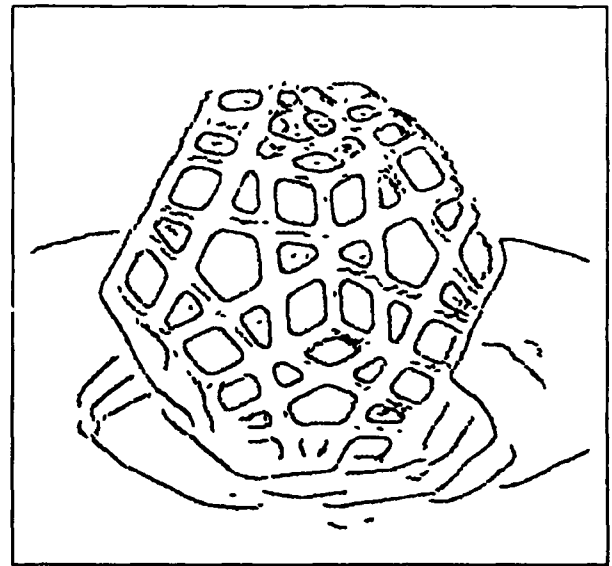
The surface being reconstructed from a moving camera can be parametrized in a natural way by two families of curves [Giblin and Weiss, 1987; Cipolla and Blake, 1990]: one family consists of the critical sets on the surface; the other is tangent to the family of rays from the camera focal points. The latter curves are called *epipolar curves*. The problem is that any smooth surface reconstruction algorithm which is more than a first order approximation requires at least three images and, that in general, the three corresponding tangent rays will not be coplanar. However, there are many cases when this will be a good approximation. One such case is when the camera trajectory is almost linear.

Cipolla and Blake [1990; 1992] and Vaillant and Faugeras [1990; 1992] noticed that to compute the curvature of a planar curve from three tangent rays, one can determine a circle which is tangent to these rays. The assumption that one needs to make is that the surface remains on the same side of the tangent rays. This is true for intervals of the curve which do not have inflections. Note that for the reconstruction of opaque surfaces, the epipolar curve on the surface ends at an inflection because the critical set disappears from view. This generally corresponds to a cusp of the profile. In addition, the epipolar curve can end where the normal to the profile is parallel to the instantaneous axis of rotation or where the critical set is occluded as at a T-junction [Giblin and Weiss, 1993].

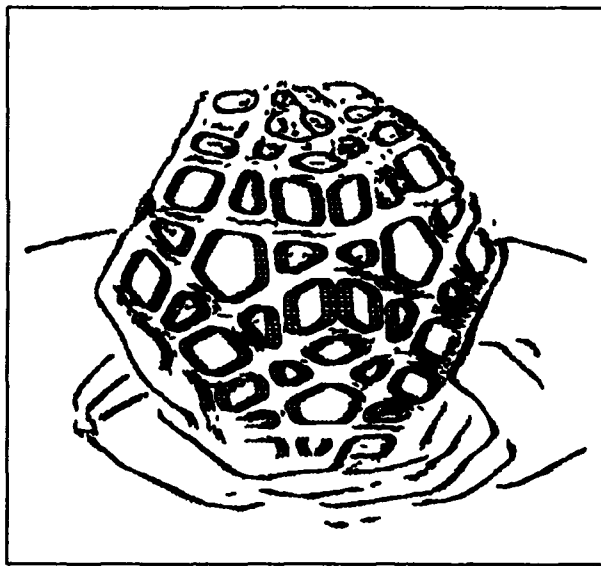
Given three or more edgels tracked with our technique, we would like to compute the location of the surface and its curvature by fitting a circular arc to the lines defined by the view directions at those edgels. In general, a space curve will have a unique circle which is closest to the curve at any given point. This is called the *osculating circle*, and the plane of this circle is called the



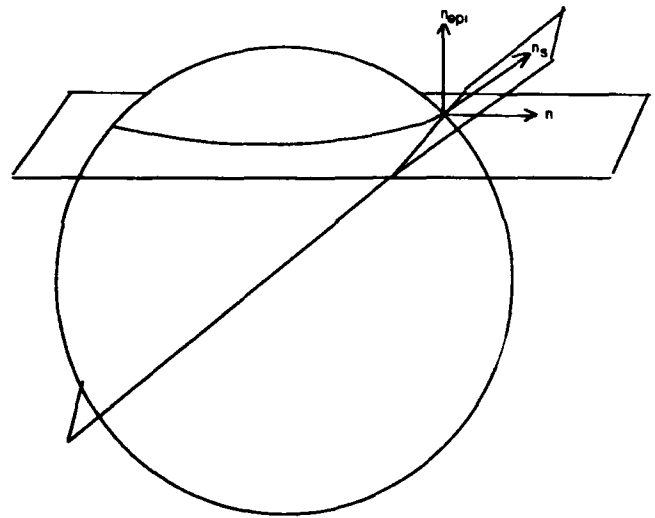
(a)



(b)



(c)



(d)

Figure 1: Input processing: (a) sample input image (dodecahedral puzzle), (b) estimated edgels and orientations (maxima in $|G_1|^2$), (c) tracked edgels, (d) correspondence of points on the occluding contours using the epipolar constraint.

While there is still some correlation between x_s and r , the estimate for x_s is much more reliable (Section 5.1). Once we have estimated (x_s, y_s, r) , we can convert this estimate back to a 3D surface point,

$$\mathbf{p}_0 = \mathbf{q}_0 + x_s \hat{\mathbf{n}}_0 + y_s \hat{\mathbf{t}}_0, \quad (6)$$

a 3D center point

$$\mathbf{c} = \mathbf{q}_0 + (x_s - r) \hat{\mathbf{n}}_0 + y_s \hat{\mathbf{t}}_0 = \mathbf{p}_0 - r \hat{\mathbf{n}}_0, \quad (7)$$

or a surface point in some other frame i

$$\mathbf{p}_i = \mathbf{c} + r \hat{\mathbf{n}}_i = \mathbf{p}_0 + r(\hat{\mathbf{n}}_i - \hat{\mathbf{n}}_0), \quad (8)$$

where

$$\hat{\mathbf{n}}_i = \hat{\mathbf{t}}_i \times \hat{\mathbf{n}}_{\text{epi}}$$

is the osculating circle normal direction perpendicular to line l_i (Figure 2).

5.1 Statistical interpretation

The least squares estimate is also the *minimum variance* and *maximum likelihood* estimate (optimal statistical estimate) under the assumption that each measurement is contaminated with additive Gaussian noise [Bierman, 1977]. If each measurement has a different variance σ_i^2 , we must weight each term in the squared error measure (2) by $w_i = \sigma_i^{-2}$, or, equivalently, multiply each equation $\mathbf{a}_i \cdot \mathbf{x} = d_i$ by σ_i^{-1} .

In our application, the variance of d_i , σ_i^2 , can be determined by analyzing the edge detector output and computing the angle between the edge orientation and the epipolar line

$$\sigma_i^2 = \sigma_e^2 / (\hat{\mathbf{l}}_i \cdot \hat{\mathbf{n}}_{\text{epi}})^2 = \sigma_e^2 / (1 - (\hat{\mathbf{m}}_i \cdot \hat{\mathbf{n}}_{\text{epi}})^2),$$

where σ_e is the variance of q_i along the surface normal $\hat{\mathbf{m}}_i$. This statistical model makes sense if the measurements d_i are noisy and the other parameters (c_i , s_i) are noise-free. This is a reasonable assumption in our case, since the camera positions are known but the edgel locations are noisy. The generalization to uncertain camera locations is left to future work.

When using least squares, the covariance matrix of the estimate can be computed from $\mathbf{P} = (\mathbf{A}^T \mathbf{A})^{-1}$. We can perform a simple analysis of the expected covariances for n measurements spaced θ apart. Using Taylor series expansions for $c_i = \cos \theta_i$ and $s_i = \sin \theta_i$, and assuming that $i \in [-m \dots m]$, $n = 2m + 1$, we obtain the covariance matrices

$$\mathbf{P}_3^c = \begin{bmatrix} 6\theta^{-4} & 0 & -6\theta^{-4} \\ 0 & \frac{1}{2}\theta^{-2} & 0 \\ -6\theta^{-4} & 0 & 6\theta^{-4} \end{bmatrix}$$

and

$$\mathbf{P}_3^s = \begin{bmatrix} 1 & 0 & -2\theta^{-2} \\ 0 & \frac{1}{2}\theta^{-2} & 0 \\ -2\theta^{-2} & 0 & 6\theta^{-4} \end{bmatrix}$$

where \mathbf{P}_3^c is the 3 point covariance for the center-point formulation, and \mathbf{P}_3^s is the 3 point covariance for the surface-point formulation. As we can see, variance of the surface point local x estimate is four orders of magnitude smaller than that of the center point. Similar results hold for the overdetermined case ($n > 3$). Extending the analysis to the asymmetrical case, $i \in [0 \dots 2m]$, we observe that the variance of the x_s and y_s estimates increases.

5.2 Robustifying the estimate

To further improve the quality and reliability of our estimates, we can apply *robust statistics* to reduce the effects of *outliers* (grossly erroneous measurements) [Huber, 1981]. Many robust techniques are based on first computing *residuals*, $r_i = d_i - \mathbf{a}_i \cdot \mathbf{x}$, and then re-weighting the data by a monotonic function

$$(\sigma'_i)^{-2} = \sigma_i^{-2} g(|r_i|)$$

or throwing out measurements whose $|r_i| \gg \sigma_i$. Alternatively, least median squares can also be used to compute a robust estimate, but at an increased complexity.

In our application, outliers occur mainly from gross errors in edge detection (e.g., when adjacent edges interfere) and from errors in tracking. Currently, we compute residuals after each batch fit, and keep only those measurements whose residuals fall below a fixed threshold.

6 Kalman filter

The Kalman filter is a powerful technique for efficiently computing statistically optimal estimates of time-varying processes from series of noisy measurements [Gelb, 1974; Bierman, 1977; Maybeck, 1979]. In computer vision, the Kalman filter has been applied to diverse problems such as motion recovery [Rives *et al.*, 1986], multiframe stereo [Matthies *et al.*, 1989], and pose recovery [Lowe, 1991]. In this section, we develop a Kalman filter for contour-based shape recovery in two parts: first, we show how to perform the batch fitting of the previous section incrementally; second, we show how surface point estimates can be predicted from one frame (and reconstruction plane) to another.

The update part of the Kalman filter is derived directly from the measurement equation (1) [Gelb, 1974]. It provides an incremental technique for estimating quantities in a *static* system, e.g., for refining a set of (x_c, y_c, r) measurements as more edgels are observed. For our application, however, we need to produce a series of surface points which can be linked together into a complete surface description. If we were using batch fitting, we would perform a new batch fit centered around each new 2D edgel. Instead, we use the complete Kalman filter, since it has a much lower computational complexity. The Kalman filter provides a way to deal with *dynamic* systems where the state \mathbf{x}_i is evolving over time. We identify each measurement \mathbf{x}_i with the surface point (x_s, y_s, r) whose local coordinate frame is given by $(\hat{\mathbf{n}}_i, \hat{\mathbf{t}}_i, \hat{\mathbf{n}}_{\text{epi}})$ centered at \mathbf{q}_i in frame i . The equations for the prediction part of the Kalman filter are derived from the mapping equations between frames (8) [Gelb, 1974].

The overall sequence of processing steps is therefore the following. Initially, we perform a batch fit to $n \geq 3$ frames, using the last frame as the reference frame. Next, we convert the local estimate into a global 3D position (6) and save it as part of our final surface model. Then, we project the 3D surface point and its radius onto the next frame, i.e., into the frame defined by the next 2D edgel found by the tracker.⁴ Then, we update the state

⁴For even higher accuracy, we could use the 2D projection of our 3D surface point as the input to our tracker.

estimate using the local line equation and the Kalman filter updating equations. We repeat the above process (except for the batch fit) so long as a reliable track is maintained (i.e., the residuals are within an acceptable range). If the track disappears or a robust fit is not possible, we terminate the recursive processing and wait until enough new measurements are available to start a new batch fit.

7 Linear smoothing

The Kalman filter is most commonly used in control systems applications, where the current estimate is used to determine an optimal control strategy to achieve a desired system behavior [Gelb, 1974]. In certain applications, however, we may wish to refine old estimates as new information arrives, or, equivalently, to use "future" measurements to compute the best current estimate. Our shape recovery application falls into this latter category, since we wish to obtain the most accurate estimate possible for the complete surface and not just the 3D curve corresponding to the currently visible occluding contour.

The generalization of the Kalman filter to update previous estimates is called the *linear smoother* [Gelb, 1974]. The smoothed estimate of x_i based on all the measurements between 0 and N is denoted by $\hat{x}_{i|N}$. Three kinds of smoothing are possible [Gelb, 1974]. In *fixed-interval smoothing*, the initial and final times 0 and N are fixed, and the estimate $\hat{x}_{i|N}$ is sought, where i varies from 0 to N . In *fixed-point smoothing*, i is fixed and $\hat{x}_{i|N}$ is sought as N increases. In *fixed-lag smoothing*, $\hat{x}_{N-L|N}$ is sought as N increases and L is held fixed.

For surface shape recovery, both fixed-interval and fixed-lag smoothing are of interest. Fixed-interval smoothing is appropriate when shape recovery is performed off-line from a set of predetermined measurements. The results obtained with fixed-interval smoothing should be identical to those obtained with a series of batch fits, but at a much lower computational cost. The fixed-interval smoother requires a small amount of overhead beyond the regular Kalman filter in order to determine the optimal combination between the outputs of a forward and backward Kalman filter [Gelb, 1974; Bierman, 1977].

For our contour-based shape recovery algorithm, we have developed a new fixed-lag smoother, which, while sub-optimal, fits in naturally with the batch and Kalman filter approaches developed in the previous two sections. Our fixed-lag smoother begins by computing a *centered* batch fit to $n \geq 3$ frames. The surface point is then predicted from frame $i-1$ to frame i as with the Kalman filter, and a new measurement from frame $i+L$, $L = \lfloor n/2 \rfloor$ is added to the predicted estimate. The addition of measurements ahead of the current estimate is straightforward using the projection equations for the least-squared (batch) fitting algorithm.

The batch fitting, Kalman filter, and linear smoothers all produce a series of surface point estimates, one for each input image. Because our reconstruction takes place in object space, features such as surface marking and sharp ridges are stationary in 3D (and have $r = 0$).

For these features, we would prefer to produce a single time-invariant estimate. While the detection of stationary features could be incorporated into the Kalman filter or smoother itself, we currently defer this decision to a post-processing stage, since we expect the estimates of position and radius of curvature to be more reliable after the whole sequence has been processed. The post-processing stage collapses successive estimates which are near enough in 3D (say, less than the spacing between neighboring sample points on the 3D contour). It adjusts the neighbor (contour) and temporal (previous/next) pointers to maintain a consistent description of the surface.

8 Experimental results

To determine the performance of our shape reconstruction algorithm, we generated a synthetic motion sequence of a truncated ellipsoid rotating about its z axis (Figure 3). The camera is oblique (rather than perpendicular) to the rotation axis, so that the motion of the pixels is not one-dimensional, and the reconstruction plane is continuously varying over time. We chose to use a truncated ellipsoid since it is easy to analytically compute its projections (which are ellipses, even under perspective), and since its radius of curvature is continuously varying (unlike, say, a sphere or a cylinder).

When we run these edge images through our least-squares fitter or Kalman filter/smoothing, we obtain a series of 3D curves. The curves corresponding to the surface markings and ridges (where the ellipsoid is truncated) should be stationary and have 0 radius, while the curve corresponding to the occluding contour should continuously sweep over the surface.

We can observe this behavior using a three-dimensional graphics program we have developed for displaying the reconstructed geometry. This program allows us to view a series of reconstructed curves either sequentially (as an animation) or concurrently (overlayed in different colors), and to vary the 3D viewing parameters either interactively or as a function of the original camera position for each frame. Figure 4 shows all of the 3D curves overlayed in a single image. As we can see, the 3D surface is reconstructed quite well. The left hand pair of images shows an oblique and top view of a noise-free data set, using the linear smoother with $n = 7$ window size. The right-hand pair shows the same algorithm with $\sigma_i = 0.1$ pixels noise added to the edge positions.

To obtain a quantitative measure of the reconstruction algorithm performance, we can compute the root median square error between the reconstructed 3D coordinates and the true 3D coordinates (which are known to the synthetic sequence generating program). Table 1 shows the reconstruction error and percentage of surface points reconstructed as a function of algorithm choice and various parameter settings. The table compares the performance of a regular 3-point fit with a 7-point moving window (batch) fit, and a linear fixed-lag smoother (labeled Kalman) with $n = 7$. Results are given for the noise-free and $\sigma_i = 0.1$ pixels case. The different columns show how by being more selective about which

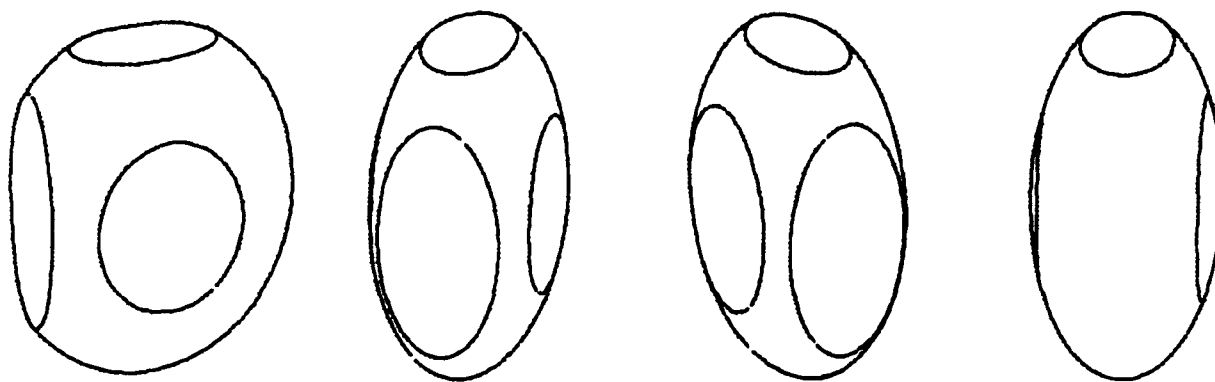


Figure 3: Four images from synthetic truncated ellipsoid sequence. The top and left hand side are truncated (cut off), while the front and back sides are inscribed with an ellipse (surface marking).

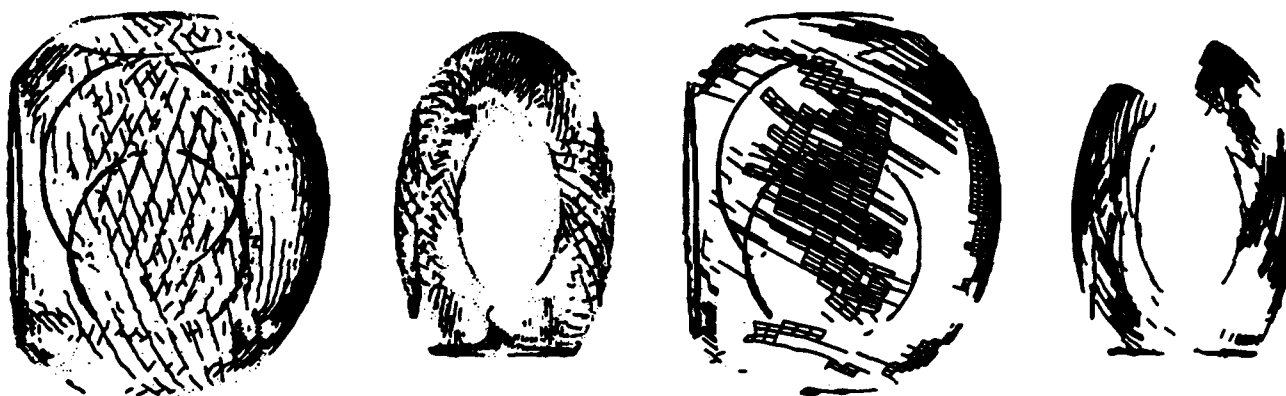


Figure 4: Oblique and top view of reconstructed 3D surface (all 3D curves are superimposed). The left pair shows only the reconstructed profile curves, while the right pair shows the profiles linked by the epipolar curves (only a portion of the complete meshed surface is shown for clarity). A total of 72 images spaced 5° apart were used.

algorithm	n	σ_i	$n_f \geq 3$	$n_f \geq 7$	$n_f \geq 7 \wedge \sigma_z^2 < 0.5$
Kalman	7	0.0	.0074 (77%)	.0046 (45%)	.0044 (38%)
Kalman	7	0.1	.0114 (74%)	.0054 (41%)	.0051 (36%)
batch	7	0.0	.0042 (79%)	.0036 (56%)	.0035 (43%)
batch	7	0.1	.0074 (77%)	.0054 (53%)	.0051 (42%)
batch	3	0.0	.0008 (77%)		
batch	3	0.1	.0159 (75%)		

Table 1: Root median square error and percentage of edges reconstructed for different algorithms, window sizes (n), input image noise σ_i , and criteria for valid estimates (n_f : minimum number of frames in fit, σ_z^2 : covariance in local z estimate). These errors are for an ellipse whose major axes are (0.67, 0.4, 0.8) and for a 128×120 image.

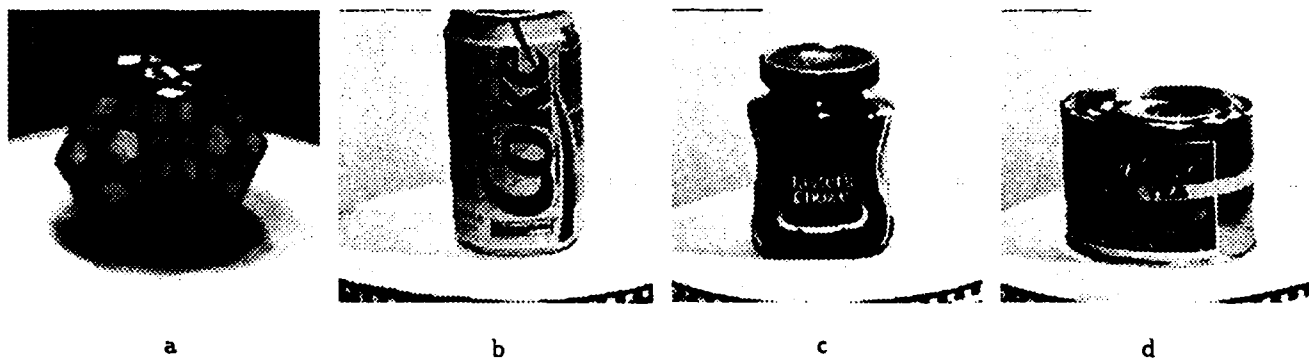


Figure 5: Sample real image sequences used for experiments: (a) dodecahedron (b) diet coke (c) coffee (d) tea.

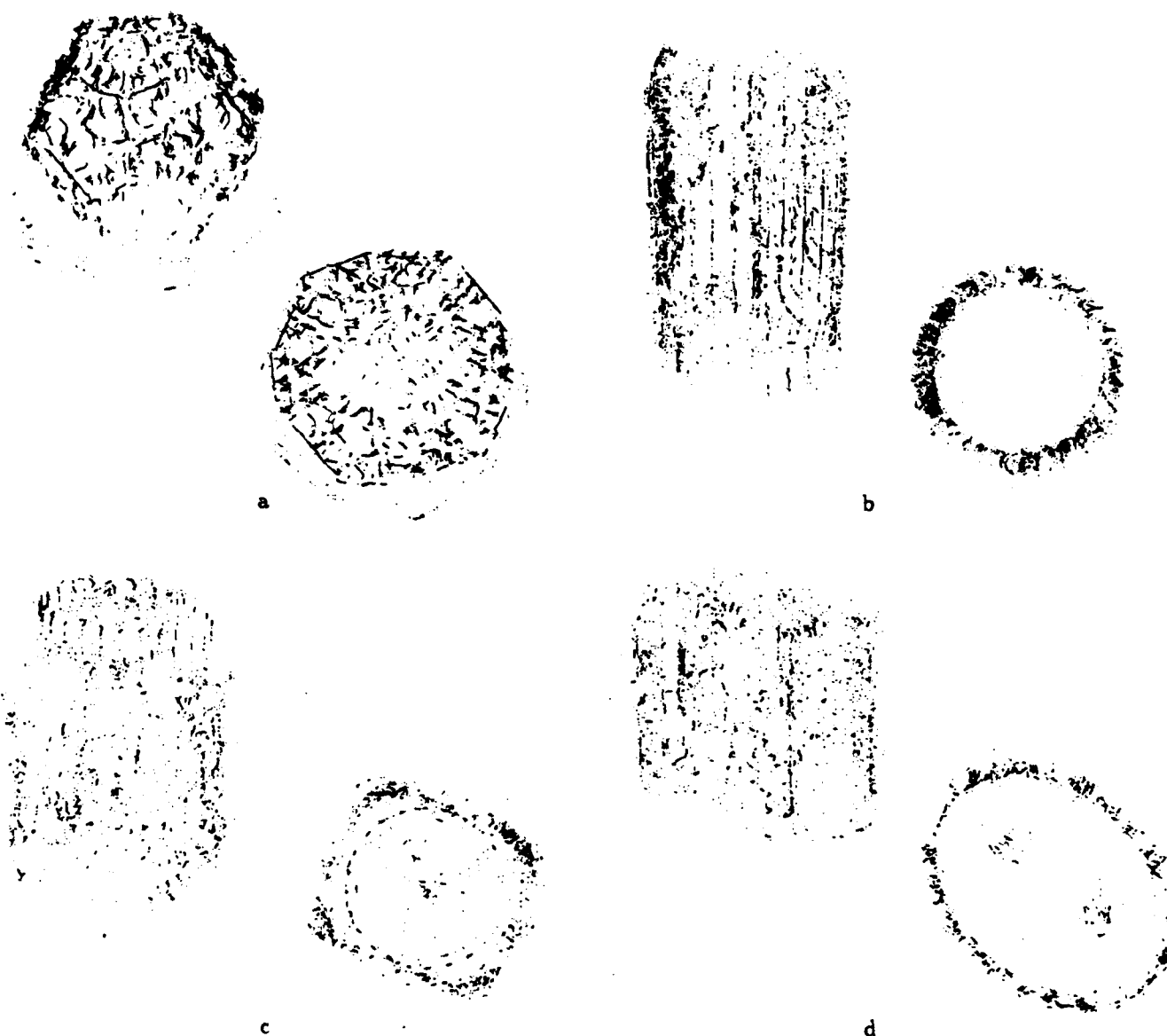


Figure 6: Oblique and top views of the 3D reconstructed curves from (a) dodecahedron, (b) diet coke, (c) coffee, and (d) tea.

3D estimates are considered valid (either by requiring more frames to have been successfully fit, or lowering the threshold on maximum covariance), a more reliable estimate can be obtained at the expense of fewer recovered points.

We have also applied our algorithm to the four real image sequences shown in Figure 5. These sequences were obtained by placing an object on a rotating mechanized turntable whose edge has a Gray code strip used for reading back the rotation angle [Szeliski, 1991]. The camera motion parameters for these sequences were obtained by first calibrating the camera intrinsic parameters and extrinsic parameters to the turntable top center, and then using the computed turntable rotation.

Figure 6 shows two views of each set of reconstructed 3D curves. We can see that the overall shape of the objects has been reconstructed quite well. We show only the profile curves, since the epipolar curves would make the line drawing too dense for viewing at this resolution.

9 Discussion and Conclusion

This paper extends previous work on both the reconstruction of smooth surfaces from profiles and on the epipolar analysis on spatiotemporal surfaces. The ultimate goal of our work is the construction a complete detailed geometric and topological model of a surface from a sequence of views. Towards this end, our observations are connected by tracking edges over time as well as linking neighboring edges into contours. The information represented at each point includes the position, surface normal, and curvatures (currently only in the viewing direction). In addition, error estimates are also computed for these quantities. Since the sensed data does not provide a complete picture of the surface, e.g., there can be self-occlusion or parts may be missed due to coarse sampling, it is necessary to build partial models. In the context of active sensing and real-time reactive systems, the reconstruction needs to be incremental as well.

Because our equations for the reconstruction algorithm are linear with respect to the measurements, it is possible to apply statistical linear smoothing techniques, as we have demonstrated. This satisfies the requirement for incremental modeling, and provides the error estimates which are needed for integration with other sensory data, both visual and tactile. The application of statistical methods has the advantage of providing a sound theoretical basis for sensor integration and for the reconstruction process in general [Szeliski, 1989; Clark and Yuille, 1990].

In future work, we intend to develop a more complete and detailed surface model by combining our technique with regularization-based curve and surface models. We also plan to investigate the integration of our edge-based multiframe reconstruction technique with other visual and tactile techniques for shape recovery.

References

- [Albert, 1972] A. Albert. *Regression and the Moore-Penrose Pseudoinverse*. Academic Press, New York, 1972.
- [Arbogast and Mohr, 1992] E. Arbogast and R. Mohr. An egomotion algorithm based on the tracking of arbitrary curves. In *Second European Conference on Computer Vision (ECCV'92)*, pages 467-475, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag.
- [Arnold, 1983] R. D. Arnold. Automated stereo perception. Technical Report AIM-351, Artificial Intelligence Laboratory, Stanford University, March 1983.
- [Baker and Bolles, 1989] H. H. Baker and R. C. Bolles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. *International Journal of Computer Vision*, 3(1):33-49, 1989.
- [Baker, 1989] H. H. Baker. Building surfaces of evolution: The weaving wall. *International Journal of Computer Vision*, 3(1):50-71, 1989.
- [Bierman, 1977] G. J. Bierman. *Factorization Methods for Discrete Sequential Estimation*. Academic Press, New York, New York, 1977.
- [Blake and Cipolla, 1990] A. Blake and R. Cipolla. Robust estimation of surface curvature from deformation of apparent contours. In *First European Conference on Computer Vision (ECCV'90)*, pages 465-474, Antibes, France, April 23-27 1990. Springer-Verlag.
- [Bolles et al., 1987] R. C. Bolles, H. H. Baker, and D. H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision*, 1:7-55, 1987.
- [Canny, 1986] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679-698, November 1986.
- [Cipolla and Blake, 1990] R. Cipolla and A. Blake. The dynamic analysis of apparent contours. In *Third International Conference on Computer Vision (ICCV'90)*, pages 616-623, Osaka, Japan, December 1990. IEEE Computer Society Press.
- [Cipolla and Blake, 1992] R. Cipolla and A. Blake. Surface shape from the deformation of apparent contours. *International Journal of Computer Vision*, 9(2):83-112, November 1992.
- [Clark and Yuille, 1990] J. J. Clark and A. L. Yuille. *Data Fusion for Sensory Information Processing Systems*. Kluwer Academic Publishers, Boston, Massachusetts, 1990.
- [Curwen et al., 1992] R. Curwen, A. Blake, and A. Zisserman. Real-time visual tracking for surveillance and path planning. In *Second European Conference on Computer Vision (ECCV'92)*, pages 879-883, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag.
- [Freeman and Adelson, 1991] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891-906, September 1991.
- [Gelb, 1974] Arthur Gelb, editor. *Applied Optimal Estimation*. MIT Press, Cambridge, Massachusetts, 1974.

- [Giblin and Weiss, 1987] P. Giblin and R. Weiss. Reconstruction of surfaces from profiles. In *First International Conference on Computer Vision (ICCV'87)*, pages 136-144, London, England, June 1987. IEEE Computer Society Press.
- [Giblin and Weiss, 1993] P. J. Giblin and R. S. Weiss. Reconstructing surfaces from known camera motion. (in preparation) 1993.
- [Giblin et al., 1992] P. J. Giblin, J. E. Rycroft, and F. E. Pollock. Moving surfaces. In *Mathematics of Surfaces V, Inst of Math and its Applications Conference*. Cambridge University Press, Sept 1992.
- [Huber, 1981] P. J. Huber. *Robust Statistics*. John Wiley & Sons, New York, New York, 1981.
- [Lowe, 1991] D. G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441-450, May 1991.
- [Marr and Hildreth, 1980] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London*, B 207:187-217, 1980.
- [Marr and Poggio, 1979] D. C. Marr and T. Poggio. A computational theory of human stereo vision. *Proceedings of the Royal Society of London*, B 204:301-328, 1979.
- [Matthies et al., 1989] L. H. Matthies, T. Kanade, and R. Szeliski. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3:209-236, 1989.
- [Maybeck, 1979] P. S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, New York, New York, 1979.
- [Mayhew and Frisby, 1980] J. E. W. Mayhew and J. P. Frisby. The computation of binocular edges. *Perception*, 9:69-87, 1980.
- [Press et al., 1986] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, 1986.
- [Rives et al., 1986] P. Rives, E. Breuil, and B. Espiau. Recursive estimation of 3D features using optical flow and camera motion. In *Conference on Intelligent Autonomous Systems*, pages 522-532. Elsevier Science Publishers, December 1986. Also appeared in 1987 IEEE International Conference on Robotics and Automation.
- [Szeliski, 1989] R. Szeliski. *Bayesian Modeling of Uncertainty in Low-Level Vision*. Kluwer Academic Publishers, Boston, Massachusetts, 1989.
- [Szeliski, 1991] R. Szeliski. Shape from rotation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'91)*, pages 625-630, Maui, Hawaii, June 1991. IEEE Computer Society Press.
- [Vaillant and Faugeras, 1992] R. Vaillant and O. D. Faugeras. Using extremal boundaries for 3-D object modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):157-173, February 1992.
- [Vaillant, 1990] R. Vaillant. Using occluding contours for 3D object modeling. In *First European Conference on Computer Vision (ECCV'90)*, pages 454-464, Antibes, France, April 23-27 1990. Springer-Verlag.

Section XV

Depth Analysis

*Integrating Multiple Range Images Using Triangulation**

Yang Chen and Gérard Medioni

Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273
E-mail: yangchen@iris.usc.edu

Abstract

We address the problem of constructing an integrated surface description of an existing object from multiple range images. The two main problems that need to be solved in such a task are integration and description. We propose to use surface triangulation as our representation for object description and data integration, since we believe that intermediate level representations, such as planar surface patches, are very flexible for shape description and their construction is local. We start from an triangulated shell and project the shell onto the range data points. An integrated approximation error measure is introduced to effectively evaluate triangle approximation error in the context of data integration. An iterative subdivision process is then applied to improve the approximation error of the triangulation to the desired precision. Test results on real range data are shown.

1 Introduction

Our goal is to construct an integrated surface description of an existing object from multiple unregistered range images. The two main problems that need to be solved in such a task are integration and description. Most of the previous research deals with the issues in description, that is, the choice of representation, how to achieve it, etc. Less work has been done on sensor data integration. It is our belief that these problems should be dealt with together, since how we perform data integration directly affects our choice of representation schemes, and thus the capability of the representation in describing objects.

Data integration can be achieved at the pixel level relatively easily, once we have all the range image views registered [Chen and Medioni 1992]. But the representations that can be used (an image parameterized in

spherical coordinates, for example) are very restricted, so that only compact (star-shaped) objects can be represented. Integration can also be done at a high level, after description of each view has been obtained [Parvin and Medioni 1991]. The disadvantage of performing high level integration is that accuracy may be sacrificed since we lose information when we go to a higher level. In addition, building descriptions from each individual view is more difficult than if complete data were available, since single views are inherently ambiguous due to self-occlusion and noise.

Therefore we believe that integration should be performed at a relative low level with a flexible representation. We have found approximation by triangles (we will call it surface triangulation, or simply triangulation hereafter) to be a good candidate for such tasks, because it is a relatively low level representation and its construction is local. A triangulated surface model can represent a variety of solid objects, and theoretically to any kind of resolution. We certainly understand the limitations of a triangulated representation. It is not ideal for high level vision tasks, such as recognition, because, first, the representation is still at low level, second, it is sensitive to many parameters, and therefore unstable. However, we think that it is a good intermediate representation for integration and for building high level description through surface interpolation from triangulation [Peters 1990].

In this paper we present a new approach to object description using multiple range images by triangulation, with emphasis on integration. We start with a triangulated shell and map it onto the object surface. The triangular approximation is refined by a triangle subdivision process. In the rest of this paper, we first review some previous research on object description using triangulation (section 2). Then we introduce the mapping by projection method (section 3), followed by approximation error estimation scheme (section 4). We present some test results on real data in section 6 and discuss the some future extensions to our system in section 7.

* This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-90-C-0078. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

2 Triangulated Representation and Triangulated Shell

In previous work, researchers have addressed the problem of achieving triangular approximations of single view range images ([De Floriani and Puppo 1988],[Schmitt and Chen 1991]). The main idea is to extend some existing 2-D triangulation techniques to find 3-D triangulations that approximate arbitrary 3-D surfaces.

Busch [Busch 1989] builds a triangulated surface of an entire object from its voxel representation by an adaptive, locally controlled growing process, which is mainly heuristically driven.

Soucy and Laurendeau [Soucy and Laurendeau 1992] use triangulation to describe objects from multiple range images. In their approach, the registered range image views are first triangulated and then integrated to form the so-called canonical views. Then the canonical views are further triangulated and the results merged by deleting overlaps and closing gaps. Their approach requires that the range image views be segmented along depth discontinuities before being triangulated.

Building a triangulation of an object can also be considered as mapping of a triangulated mesh onto the surface of the object, so that the triangles locally approximate the surface. Instead of building pieces of the triangulation and then putting them together, we can also start with a closed surface tessellated with triangles, or any suitable primitive surface patches in general, and map it onto (or deform it so as to match) the surface. This way we can avoid much of the heuristic approach in dealing with growing the triangulation or merging the pieces of triangulation. The adaptive shell proposed by Vasilescu and Terzopoulos [Vasilescu and Terzopoulos 1992] and the deformable surface by Delingette *et al.* [Delingette *et al.* 1991] are two examples of such an approach.

There are two issues we need to consider in such a mapping. The first is the capability of the mapping in handling objects with complex shape (e.g., non-star-shaped objects). The second is the computational complexity and convergence. The two aforementioned methods all use a dynamic shape model which incorporates internal surface forces and external data forces. In principle, these methods are able to describe complex objects. The difficulty lies in defining a good external energy function that can be both easy to compute and accurate in reflecting the fit of the shape model to the data. There are also problems in selecting an initial surface and in dealing with local minima. The computational cost is also very high for such dynamic systems. In addition, their approach does not address the problem of integration, which is one of our goals. We also try to

achieve such a mapping without solving large dynamic systems.

3 Mapping the Triangulated Shell by Projection

Here we present a new method for mapping the triangulated shell to the object surface. In this method, a triangulated shell is first initialized around the object. Then the triangles are mapped onto the object surface by projecting them in the radial direction from the center of the projection. The projection is obtained by computing the intersections of the lines of projection and the object surface represented by the range images. We first present the details of the method and then discuss the advantages and disadvantages of the method.

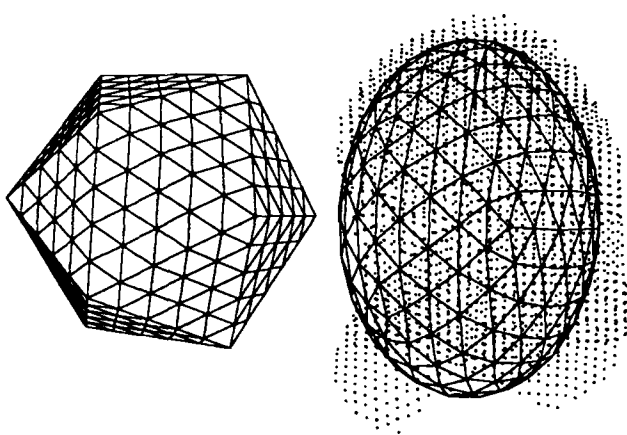
3.1 Input Data

Our input is a set of range image views of an object acquired with a liquid-crystal range finder [Sato and Inokuchi 1987]. In a previous paper [Chen and Medioni 1992], we have presented a method to register range images of an object, which can find registration transformation between range images of the same object from different views. We then register the acquired views and record the global transformation information for each view. Although registered, the individual views might not align exactly everywhere, partly reflecting some defects present in the raw data and partly due to mis-registration caused by noise and other defects. We assume that the range data is dense enough for evaluating surface properties such as surface normals using neighborhood fitting.

3.2 Initializing and Projecting the Triangulated Shell

Our goal is to map a triangulated shell onto the surface of an object. We start with an icosahedron and divide each of its triangular faces into $N \times N$ sub-triangles [Delingette *et al.* 1991] (Figure 1. (a)), where N is the order of initial subdivision. The triangulated shell is then re-initialized into an ellipsoidal shell of approximately the size of the object (Figure 1. (b)), based on the distribution information of the sample points from the range images. This helps in finding the intersections between the line of projection and the object surface, as explained later.

The triangulated ellipsoidal shell is projected onto the object surface by projecting the vertices, from the center of the projection (see later in this section for the selection of the center). To find the projection for a vertex, a ray (the line of projection) is constructed from the center of the projection through the initial position of the vertex and the intersection between the ray and the surface is computed using an iterative algorithm [Chen and Medioni 1992]. This line-surface intersection algorithm works directly with range images without an ana-



(a) An icosahedron with each of its faces subdivided into 25 sub-triangles ($N=5$) (b) An initialized ellipsoid (with sample data superimposed)

Figure 1. Initializing the ellipsoidal shell

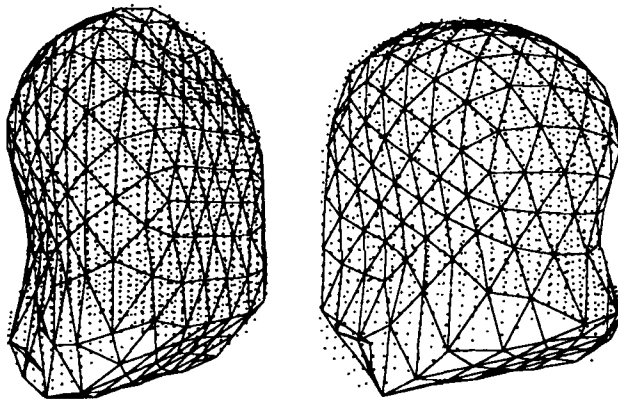


Figure 2. Result of projecting the ellipsoidal shell onto the surface of an object. Shown with the wire frames are sample surface points from the range image views of the object.

lytical representation of the surface. At each iteration, the object surface near the prospective intersection point is approximated linearly using its tangent plane, which intersects the line in consideration. This intersection converges to the intersection of the surface with the line. The starting point for such an iterative algorithm must be relatively close to the intersection point, which also means close to the surface. In our case the starting point is chosen to be initial position of the vertex itself. Thus an initial triangulated shell approximating the object surface is desirable. An example of the projection is shown in Figure 2..

In summary, what are essential in the projection are (a) the direction of the projection (in this case the radial direction), and (b) an initial point on the line of projection. As we have mentioned early, the initial ellipsoidal

shell is ideal for initializing the vertices close to the object surface, while (a) requires that a center of projection be selected. The requirements for such a point are that it lies inside of the object, and there is as much as possible surface areas visible from this point. We discuss the implications of these requirements further in section 7.

3.3 Multiple Intersections

Since the input data is in the form of multiple range images from many view points, there are overlaps in the surface areas that each range image covers. This results in multiple projection points on the range images when a vertex is projected using the method described above. Assuming that there is only one intersection between the projection ray and the object surface, we need to combine the set of intersections computed from range images of different views into one. Simple averaging of all the points is one solution, but it does not work well as it assume the noise distribution to be Gaussian. This is clearly not the case when two surface are slightly out of registration. We have adopted a weighted average method that takes into account the reliability of the input data. This is because range data from the surface areas facing the sensor are much less noisy and thus more reliable than data from areas with large incidence angles with respect to the sensor. Let $p_i \in R^3$, $i=1 \dots m$ be the intersection points of the line of projection with range image view V_i , and n_i and s_i the surface normal at p_i and the sensor direction for range image V_i , respectively. The weighted average p of the intersection points p_i is defined by

$$p = \frac{\sum_i^n (n_i \cdot s_i) p_i}{\sum_i^n n_i \cdot s_i} \quad (1)$$

where $a \cdot b$ stands for the inner product of two vectors a and b . In our experiments, the surface normal vectors are obtained from the range images by using a neighborhood surface fitting technique.

4 Approximation via Triangle Subdivision

When the triangulated shell is projected onto the object surface, each triangle may or may not approximate the covered surface well. In this section, we discuss how to evaluate the approximate error of the triangles, especially in the context of data integration, and how to improve the approximation error by subdividing the triangles.

The first problem is to define the data points that need to be considered in evaluating the approximation error for a triangle T . Naturally we do not need to consider those range images on which none of the triangle

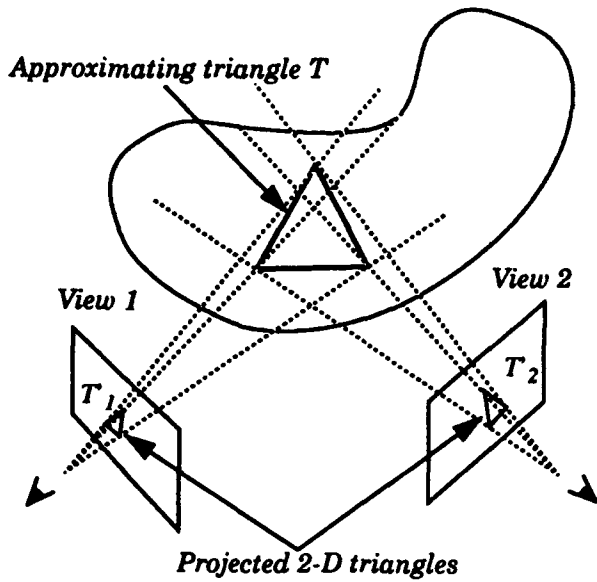


Figure 3. A 3-D triangle projected into range image parameter space.

vertices have projections. For the range images on which at least one of the vertices has projections, we project (not to be confused with the projection mentioned early) the triangle T into the range image parameter space to obtain a 2-D triangle T' , as though the triangle had been in the place when range images were taken (see Figure 3.).

In the simplest case of a Cartesian range image, where the depth value z is indexed by the x and y coordinates as $z=f(x,y)$, T' is simply the orthographic projection of T . In general, however, the projection T' of T involves a perspective transformation from 3-D world coordinates into sensor coordinate. Such a transformation is usually available from system calibration of the range finder [Sato and Inokuchi 1987]. Once the projected 2-D triangle for each view is obtained, all surface points in that view that fall inside of the 2-D triangle will be considered in evaluating the approximation error. We call those 3-D points the covered points of triangle T .

The second problem in the approximation error evaluation is to define a measure for the approximation error. In triangulating single view range images [Schmitt and Chen 1991] (or the reparameterized, combined range images as in [Soucy and Laurendeau 1992]), we could just use the maximum Euclidean distance of the covered points to the plane containing the triangle in consideration. But this will not work when there are multiple views. The reason is that although the range image views are well registered in general [Chen and Medioni 1992], some views might deviate from the rest. There can also be regions that contains contaminated data. A simple maximum distance measure does not

truly reflect the status of the approximation in such situations. Therefore we have adopted a statistical approach which allows us to eliminate the bad data points and bad range image views for each approximating triangle.

Let T'_i be the projected 2-D triangle of T for range image view V_i , $i=1 \dots m$, where m is the total number of range images that have at least one vertex of T projected on them. Let $Q_i = \{q_{ij}\}$ be the set of 3D points in V_i covered by T'_i , and e_{ij} be the signed Euclidean distance of q_{ij} to the plane containing T . We can compute the mean and standard deviation of e_{ij} :

$$\bar{e}_i = \frac{\sum_{q_{ij} \in Q_i} e_{ij}}{n_i}, \sigma_i = \sqrt{\frac{\sum_{q_{ij} \in Q_i} (e_{ij} - \bar{e}_i)^2}{n_i}} \quad (2)$$

where $n_i = \|Q_i\|$ is the cardinality of the set Q_i . Then we adopt a majority vote scheme to effectively eliminate the bad views from being considered in the final approximation error estimation for that triangle. That is, we compute a subset $G \subseteq \{Q_i, i=1 \dots m\}$, such that the absolute value of the difference of the mean approximation error \bar{e}_i between any pair in G is below certain threshold:

$$G = \{Q_i \mid \forall Q_k \in G, |\bar{e}_i - \bar{e}_k| < \epsilon\} \quad (3)$$

and that the cardinality of G is at least half of the original set:

$$\|G\| \geq 0.5 \| \{Q_i\} \| \quad (4)$$

The value of ϵ can be set based on the range image resolution and the statistics of the registration errors from the registration process [Chen and Medioni 1992]. Once G is selected, we recompute the mean and standard deviation of the approximation error for all the points in G for triangle T . Let the results be \bar{e} and σ respectively. In the case when G is an empty set, or G is a minority group, then we simply select the Q_i that has the least absolute \bar{e}_i :

$$G' = \{Q_i \mid |\bar{e}_i| = \min_{Q_k \in \{Q_i\}} (|\bar{e}_k|)\} \quad (5)$$

The overall approximation measure is defined as

$$E = \bar{e} + \text{sign}(\bar{e})\alpha\sigma \quad (6)$$

where $\text{sign}()$ returns 1 or -1 depending on whether its argument is larger or smaller than 0. α is a constant that reflects the noise present in the data. The approximation measure E in equation above is used to simulate the maximum approximation error for a triangle. We have

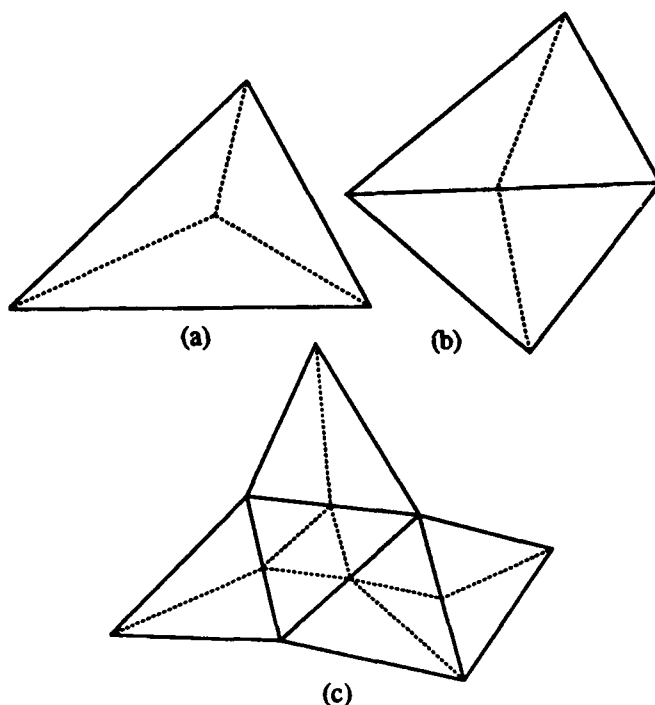


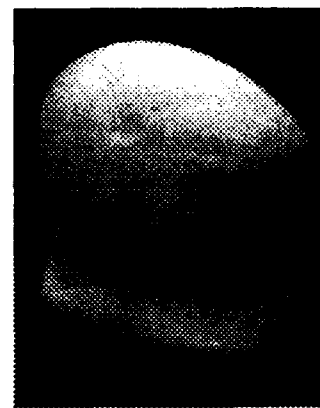
Figure 4. Triangles are subdivided in groups with the same type of approximation errors. The dashed lines define the new triangles.

found $\alpha=2$ to be a good choice, which can eliminate most of the outliers in the data.

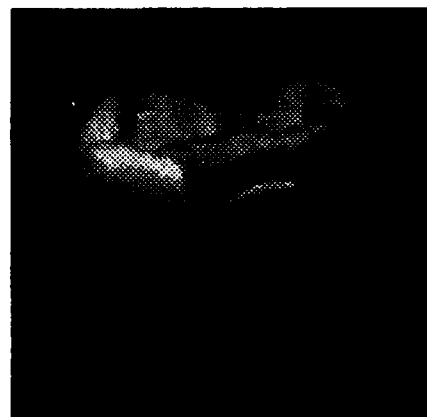
5 Triangle Subdivision and Reprojection

Once the approximation error for a projected triangle is evaluated, those triangles whose approximation error E does not meet the predefined approximation threshold must be subdivided and remapped. Our subdivision algorithm tries to subdivide triangles without altering the neighborhood adjacency property of the triangulation, i.e., each triangle must have 3 neighbors that share an edge with it. This regularity makes it easy for higher order surface reconstruction in the future. For this reason, we group triangles with the same type of approximation error (i.e., triangles with \bar{e} of the same sign and their $|\bar{e}|$ exceeding a certain given threshold) into connected components. Here, two triangles are connected when they share an edge. The subdivision method is illustrated in Figure 4., categorized by the size of the connected components.

The rule here is that, except for the components that contain only one triangle, triangles with one neighboring triangle in the connected component will be divided into two, those with two neighbors three, and those with three neighbors four. To improve the fit of the triangles to the object surface, the newly created triangles must be remapped onto the object surface. This is done by projecting the triangle vertices in the direction of the



(a) A free-style wood blob (the "wood")



(b) A plaster model tooth (the "tooth")

Figure 5. Intensity images of the test objects.

surface normals of the object at that location, which can be approximated by the average normal vector of the surrounding triangles of the vertex, weighted by the areas of those triangles. The difference between this projection and the initial projection is the direction in which the projection is done.

This division-and-projection process continues until all the triangles have an approximation error within the desired threshold.

6 Experimental Results

Now we show some test results of the triangle subdivision and approximation. Figure 5. shows two objects ("wood" and "tooth") in intensity images, which are approximately 8 and 10cm high respectively.

Range images are acquired for each object from 16 different viewpoints and then registered. Figure 6. shows four views of the range images of "free" used in the test. Figure 7. shows wire frames of the triangle subdivision process. Figure 7. (a) shows the triangulated shell projected onto the object surface. Figure 7. (b) shows the shell after one subdivision iteration with ap-



(a) Range image view 1 (b) Range image view 2



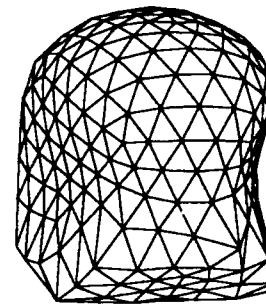
(c) Range image view 3



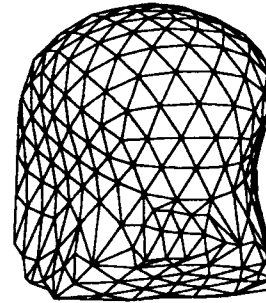
(d) Range image view 4

Figure 6. Sample range image views of the object "wood" used for model construction, shown here as shaded images.

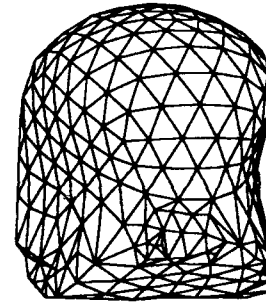
proximation error threshold set to 2mm (see Section 5 and equation (6)). Figure 7. (c) is the final result after three iterations with all triangles having approximation errors under 2mm. As can be seen, most of the subdivisions take place around the ridge line in the lower part of the object. Figure 8. and Figure 9. show some rendered intensity images of the "wood" and the "tooth" using the derived triangulation models from various view points. In both cases, the initial icosahedrons are subdivided with $N=5$ and the center of the fitting ellipsoidal shells are used as the center of the projection. Our test programs are written in Lisp. On a Sun SparcStation 2, with Sun Common Lisp, the test on the "wood" completed in 3 minutes and 18 seconds, resulting a total of 649 triangles, while the "tooth" took 3 minutes and 34 seconds yielding 872 triangles. Note that the original



(a) Before subdivision



(b) After 1 iteration



(c) After 3 iterations

Figure 7. Iterative triangle subdivision

range images have a resolution of about 1 to 2mm with a comparable registration error.

7 Working with Complex Objects

As mentioned in section 5, in projecting the initial triangulated shell, we need to select a projection center, which must be inside of the object. This is because, if the center is outside of the object, the projected triangles will not always be able to maintain their proper relationship among them and surface self intersection may occur, as shown in Figure 10. (a). Another important issue is that when the object is non-star shaped, we will have multiple intersections when performing the projection, regardless of where we put the projection center, as shown in Figure 10. (b). There are two possible ways to deal with such situations. The first is to choose the inner most (closest to the center of projection) inter-

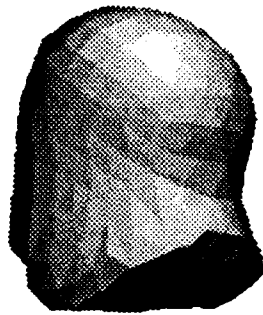


Figure 8. Two rendered views of the constructed triangulation of the object "wood"

sections (Figure 10. (b)) and the second is to choose the outer most intersections (figureFigure 10. (c)). In both cases, the problems which remain are how to identify triangles that cut through the object and how to properly group them together and find a local projection center for subdivided triangles derived from them.

8 Conclusion

We have presented a new method for constructing an integrated surface description from multiple range images based on surface triangulation. Our system combines the surface description and data integration through an effective evaluation for triangle approximation error using an integrated error measure. Projecting a triangulated shell onto the surface of the object has proven to be very efficient and effective in acquiring triangulated surface models compared to systems that employ global optimization techniques. Preliminary tests show good results on free-formed compact objects. Our future research is concerned with extending the current system to handle more complex objects.

Acknowledgments

The authors would like to thank Mr. C. Liao for the triangular subdivision algorithm presented in this paper.

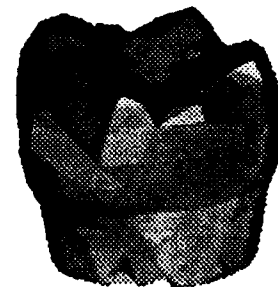
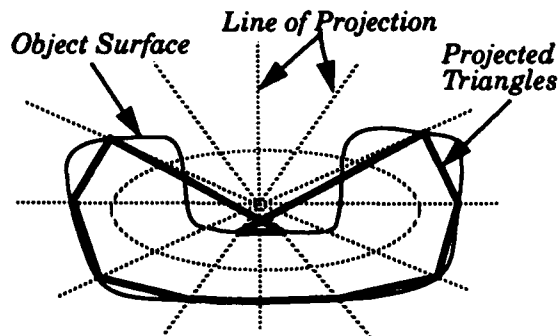


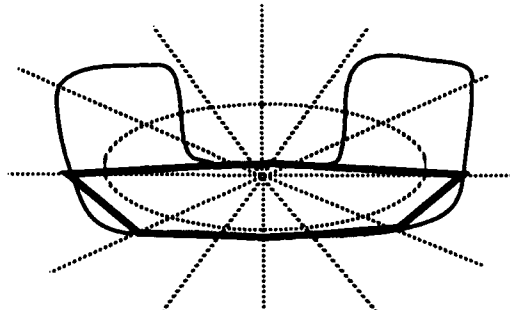
Figure 9. Three rendered views of the constructed triangulation of the object "tooth"

References

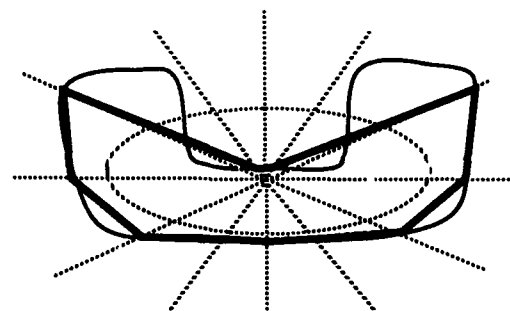
- [Busch 1989] Hans Busch, "Automatic Modeling of Rigid 3D Objects Using an Analysis by Synthesis System," In W.A. Pearlman, editor, *Visual Communications and Image Processing IV*, volume Proc. SPIE Vol.1199, pages 356-364, Philadelphia, PA, November 8-10 1989.
- [Chen and Medioni 1992] Y. Chen and G. Medioni, "Object Modelling by Registration of Multiple Range Images," *International Journal of Image and Vision Computing*, 10(3):145-155, April 1992.



(a) The center of projection is outside of the object



(b) Using innermost intersections



(c) Using outermost intersections

Figure 10. Selecting the center of projection and dealing with multiple intersections

- [Delingette et al. 1991] H. Delingette, M. Hebert, K. Ikeuchi, "Shape Representation and Image Segmentation Using Deformable Surfaces," *CVPR 1991* pp.467-472.
- [De Floriani and Puppo 1988] L. De Floriani and E. Puppo, "Constrained Delaunay Triangulation for Multiresolution Surface Description," In *Proceedings of the International Conference on Pattern Recognition*, pages 566-569, Rome Italy, Nov. 14-17 1988. Computer Society Press, Washington, DC.
- [Parvin and Medioni 1991] B. Parvin and G. Medioni, "A Dynamic System for Object Description and Correspondence," *Proc. CVPR*, Jun. 1991, Maui, Hawaii, pp. 393-399.

- [Peters 1990] J. Peters, "Local Smooth Surface Interpolation: A Classification," *Computer Aided Geometric Design*, 7:191-195, 1990.
- [Sato and Inokuchi 1987] K. Sato and S. Inokuchi, "Range-Imaging System Utilizing Nematic Liquid Crystal Mask," In *Proceedings of the IEEE International Conference on Computer Vision*, pages 657-661, London, England, June 1987.
- [Schmitt and Chen 1991] F. Schmitt and X. Chen, "Fast Segmentation of Range Images into Planar Regions," In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Maui, Hawaii, June 1991. Computer Society Press.
- [Soucy and Laurendeau 1992] M. Soucy and D. Laurendeau, "Multi-resolution surface modeling from range views," In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 348-353, Urbana-Champaign, IL, June 1992.
- [Vasilescu and Terzopoulos 1992] M. Vasilescu and Demetri Terzopoulos, "Adaptive Meshes and Shells: Irregular Triangulation, Discontinuities, and Hierarchical Subdivision," *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 829-832. Urbana-Champaign, IL, June 1992.

Range Estimation From Focus Using a Non-frontal Imaging Camera*

Arun Krishnan and Narendra Ahuja

Beckman Institute
University Of Illinois
405 North Mathews Ave.,
Urbana IL 61801, U.S.A.

e-mail: arunki@vision.csl.uiuc.edu

e-mail: ahuja@vision.csl.uiuc.edu

Abstract

This paper is concerned with active sensing of range information from focus. It describes a new type of camera whose image plane is not perpendicular to the optical axis as is standard. This special imaging geometry eliminates the usual focusing need of image plane movement. Camera movement, which is anyway necessary to process large visual fields, integrates panning, focusing, and range estimation. Thus the two standard mechanical actions of focusing and panning are replaced by panning alone. Range estimation is done at the speed of panning. An implementation of the proposed camera design is described and experiments with range estimation are reported.

1 INTRODUCTION

This paper is concerned with active sensing of range information from focus. It describes a new type of camera which integrates the processes of image acquisition and range estimation. The camera can be viewed as a computational sensor which can perform high speed range estimation over large scenes. Typically, the field of view of a camera is much smaller than the entire visual field of interest. Consequently, the camera must pan to sequentially acquire images of the visual field, a part at a time, and for each part compute range estimates by acquiring and searching images over many image plane locations. Using the proposed approach, range can be computed at the speed of panning the camera.

At the heart of the proposed design is active control of imaging geometry to eliminate the standard mechanical adjustment of image plane location, and

*The support of the National Science Foundation and Defence Advanced Research Projects Agency under grant IRI-89-02728 and U.S. Army Advance Construction Technology Center under grant DAAL 03-87-K-0006 is gratefully acknowledged.

further, integration of the only remaining mechanical action of camera panning with focusing and range estimation. Thus, imaging geometry and optics are exploited to replace explicit sequential computation. Since the camera implements a range from focus approach, the resulting estimates have the following characteristics as is true for any such approach [3, 5]. The scene surfaces of interest must have texture so image sharpness can be measured. The confidence of the estimates improves with the amount of surface texture present. Further, the reliability of estimates is inherently a function of the range to be estimated. However, range estimation using the proposed approach is much faster than any traditional range from focus approach, thus eliminating one of the major drawbacks.

The next section describes in detail the pertinence of range estimation from focus, and some problems that characterize previous range from focus approaches and serve as the motivation for the work reported in this paper. Then, Section 3 presents the new, proposed imaging geometry whose centerpiece is a tilting of the image plane from the standard frontoparallel orientation. It shows how the design achieves the results of search over focus with high computational efficiency. Section 4 presents a range from focus algorithm that uses the proposed camera. Section 5 describes the results of experiments demonstrating the feasibility of our method. Section 6 presents concluding remarks.

2 BACKGROUND & MOTIVATION

2.1 Range Estimation From Focus and Its Utility

Focus based methods obtain a depth estimate of a scene point by varying the focal-length (f) and/or the focus distance (v). Without loss of generality, we will always assume that the parameter being controlled is v . This means that the v value

is changed by mechanically relocating the image plane. When the scene point appears in sharp focus, the corresponding u and v values satisfy the standard lens law: $\frac{1}{u} + \frac{1}{v} = \frac{1}{f}$. The depth value u for the scene point can then be calculated by knowing the values of the focal length and the focus distance [14, 2, 6]. To determine when a scene is imaged in sharp focus, several autofocus methods have been proposed in the past [7, 16, 17, 8, 12, 15, 10, 9, 2, 13].

Like any other visual cue, range estimation from focus is reliable under some conditions and not so in some other conditions. Therefore, to use the cue appropriately, its shortcomings and strengths must be recognized and the estimation process should be suitably integrated with other processes using different cues, so as to achieve superior estimates under broader conditions of interest [1, 11, 4]. When accurate depth information is not needed, e.g., for obstacle avoidance during navigation, range estimates from focus or some other cue alone may suffice, even though it may be less accurate than that obtained by an integrated analysis of multiple cues.

2.2 Motivation for Proposed Approach

The usual range from focus algorithms involve two mechanical actions, those of panning and for each chosen pan angle finding the best v value. These steps make the algorithms slow. The purpose of the first step is to acquire data over the entire visual field since cameras typically have narrower field of view. This step is therefore essential to construct a range map of the entire scene. The proposed approach is motivated primarily by the desire to eliminate the second step involving mechanical control.

Consider the set of scene points that will be imaged with sharp focus for some constant value of focal length and focus distance. Let us call this set of points the SF surface¹. For the conventional case where the image is formed on a plane perpendicular to the optical axis, and assuming that the lens has no optical aberrations, this SF surface will be a surface that is approximately planar and normal to the optical axis. The size of SF surface will be a scaled version of the size of the image plane, while its shape will be the same as that of the image plane. Figure 1(a) shows the SF surface for a rectangular image plane.

As the image plane distance from the lens, v , is changed, the SF surface moves away, or toward the camera. As the entire range of v values is traversed, the SF surface sweeps out a cone shaped volume in three-dimensional space, henceforth called the

SF cone. The vertex angle of the cone represents the magnification or scaling achieved and is proportional to the f value. Figure 1(b) shows a frustum of the cone.

Only those points of the scene within the SF cone are ever imaged sharply. To increase the size of the imaged scene, the f value used must be increased. Since in practice there is a limit on the usable range of f value, it is not possible to image the entire scene in one viewing. The camera must be panned to repeatedly image different parts of the scene. If the solid angle of the cone is ω , then to image an entire hemisphere one must clearly use at least $\frac{2\pi}{\omega}$ viewing directions. This is a crude lower bound since it does not take into account the constraints imposed by the packing and tessellability of the hemisphere surface by the shape of the camera visual field.

If specialized hardware which can quickly identify focused regions in the image is used, then the time required to obtain the depth estimates is bounded by that required to make all pan angle changes and to process the data acquired for each pan angle.

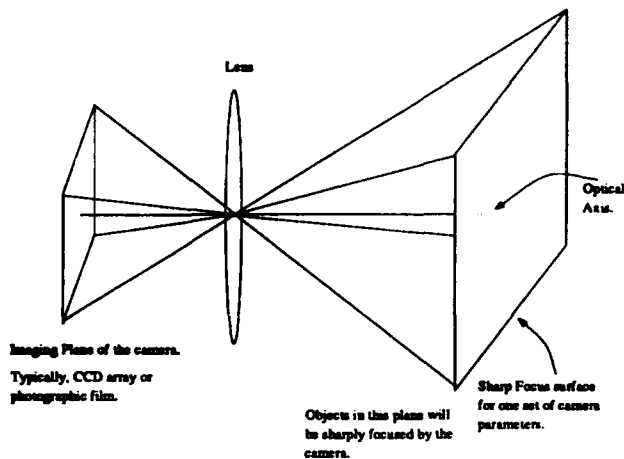
The goal of the approach proposed in this paper is to select the appropriate v value for each scene point without conducting a dedicated mechanical search over all v values. The next section describes how this is accomplished by slightly changing the camera geometry and exploiting this in conjunction with the pan motion to accomplish the same result as traditionally provided by the two mechanical motions.

3 A NON-FRONTAL IMAGING CAMERA

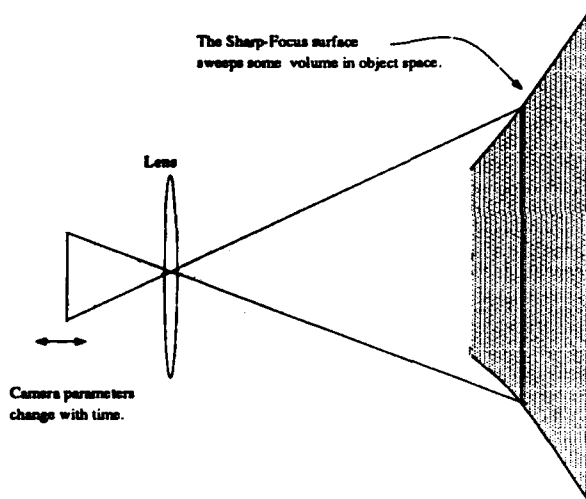
The following observations underlie the proposed approach. In a normal camera, all points on the image plane lie at a fixed distance (v) from the lens. So all scene points are always imaged with a fixed value of v , regardless of where on the image plane they are imaged, i.e., regardless of the camera pan angle. If we instead have an image surface such that the different image surface points are at different distances from the lens, then depending upon where on the imaging surface the image of a scene point is formed (i.e., depending on the pan angle), the imaging parameter v will assume different values. This means that by controlling only the pan angle, we could achieve both goals of the traditional mechanical movements, namely, that of changing v values as well as that of scanning the visual field, in an integrated way.

In the rest of this paper, we will consider the simplest case of a nonstandard image surface, namely a plane which has been tilted relative to the standard frontoparallel orientation. Consider the tilted

¹ Actually, the depth-of-field effect will cause the SF surface to be a 3-D volume. We ignore this for the moment, as the arguments being made hold irrespective of whether we have a SF surface, or a SF volume.



(a) SF surface



(b) SF cone

Figure 1: (a) Sharp Focus object surface for the standard planar imaging surface orthogonal to the optical axis. Object points that lie on the SF surface are imaged with the least blur. The location of the SF surface is a function of the camera parameters. (b) A frustum of the cone swept by the SF surface as the value of v is changed. Only those points that lie inside the SF cone can be imaged sharply, and therefore, range-from-focus algorithms can only calculate the range of these points.

image plane geometry shown in Figure 2. For different angles θ , the distance from the lens center to the image plane is different. Consider a point object at an angle θ . The following relation follows from the geometry:

$$|\vec{OC}| = \frac{d \cos \alpha}{\cos(\theta - \alpha)}$$

Since for a tilted image plane, v varies linearly with position, it follows from the lens law that the corresponding SF surface is a plane whose u value mirrors the v variation. The SF surface is shown in Figure 3(a). The volume swept by the SF surface as the camera is rotated is shown in Figure 3(b).

If the camera turns about the lens center O by an angle ϕ , then the object will now appear at an angle $\theta + \phi$. The new image distance (for the point object) will now be given by the following equation.

$$|\vec{OC}| = \frac{d \cos \alpha}{\cos(\phi + \theta - \alpha)}$$

As the angle ϕ changes, the image distance also changes. At some particular angle, the image will appear perfectly focused and as the angle keeps changing, the image will again go out of focus. By identifying the angle ϕ at which any surface appears in sharp focus, we can calculate the image distance, and then from the lens law, the object surface distance.

As the camera rotates about the lens center, new parts of the scene enter the image at the left edge² and some previously imaged parts are discarded at the right edge. The entire scene can be imaged and ranged by completely rotating the camera once.

4 RANGE ESTIMATION ALGORITHM

Let the image plane have $N \times N$ pixels and let the range map be a large array of size $N \times sN$, where $s \geq 1$ is a number that depends on how wide a scene is to be imaged. The k^{th} image frame is represented by I_k and the cumulative, environment centered range map with origin at the camera center is represented by R . Every element in the range array is a structure that contains the focus criterion values for different image indices, i.e., for different pan angles. When the stored criterion value shows a maximum³, then the index corresponding to the maximum³ can be used to determine the range for that scene point.

²Or the right edge, depending upon the direction of the rotation

³Knowing the index value, we can find out the amount of camera rotation that was needed before the scene point was sharply focused. Using the row and column indices for the range point, and the image index, we can then find out the exact distance from the lens to the image plane (v). We can then use the lens law to calculate the range.

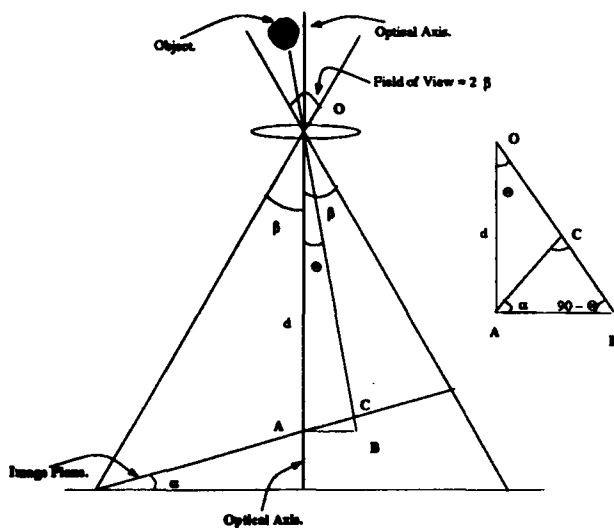


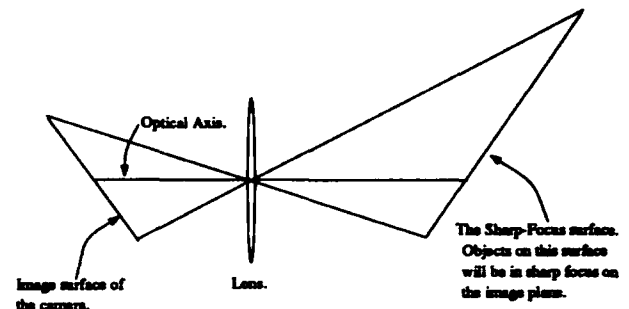
Figure 2: Tilted Image Surface. A point object, initially at an angle of θ , is imaged at point C. The focus distance OC varies as a function of θ . When the camera undergoes a pan motion, θ changes and so does the focus distance.

Let the camera start from one side of the scene and pan to the other side. Figures 4(a) and 4(b) illustrate the geometrical relationships between successive pan angles, pixels of the images obtained, and the range array elements.

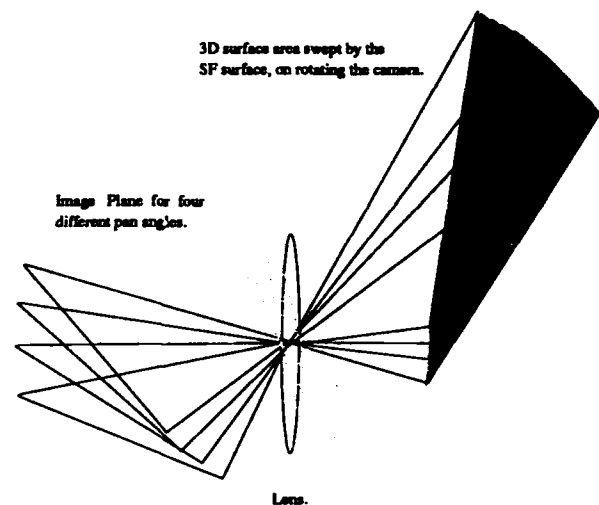
4.1 Algorithm

Let $j = 0$. $\phi = 0$. Initialize all the arrays and then execute the following steps.

- Capture the j^{th} image I_j .
- Pass the image through a focus criterion filter to yield an array C_j of criterion values.
- For the angle ϕ (which is the angle that the camera has turned from its starting position), calculate the offset into the range map required to align image I_j with the previous images. For example, Pixel $I_j[50][75]$ might correspond to the same object as pixels $I_{j+1}[50][125]$ and $I_{j+2}[50][175]$.
- Check to see if the criterion function for any scene point has crossed the maximum. If so, compute the range for that scene point using the pan angle (and hence ν value) for the image with maximum criterion value.
- Rotate the camera by a small amount. Update ϕ and j .
- Repeat the above steps until the entire scene is imaged.



(a) SF surface



(b) SF cone

Figure 3: (a) The SF surface for the proposed camera with a tilted image plane. The SF surface is not parallel to the lens and the optical axis is not perpendicular to the SF surface. (b) The 3D volume swept by the proposed SF surface as the non-frontal imaging camera is rotated. For the same rotation, a frontal imaging camera would sweep out an SF cone having a smaller depth.

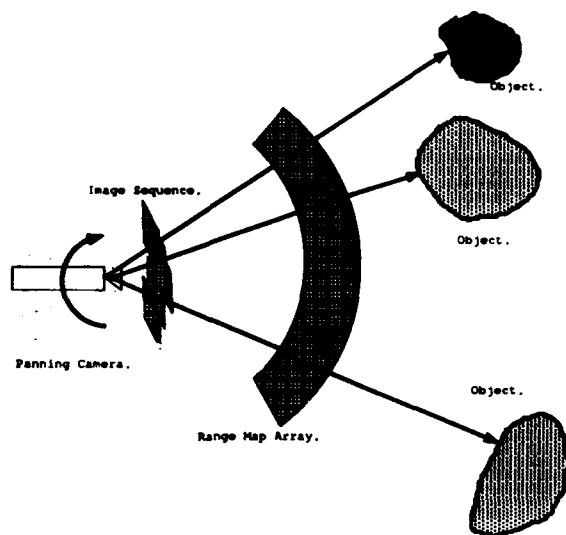
5 EXPERIMENTAL RESULTS

In these experiments, we attempt to determine the range of scene points. The scene in experiment 1 consists of, from left to right, a planar surface (range = 73 in), part of the background curtain (range = 132 in), a planar surface (range = 54 in) and a planar surface (range = 38 in). The scene in experiment 2 consists of, from left to right, a planar surface (range = 70 in), a planar surface (range = 50 in), and a face of a box (at a depth of 35 in).

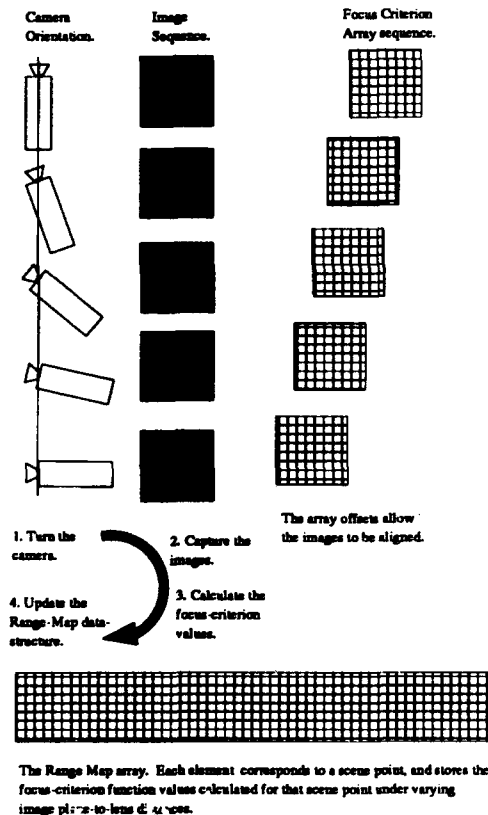
The camera is turned in small steps of 50 units (of the stepper motor), that corresponds to a shift of 15 pixels (in pixel columns) between images. A scene point will thus be present in a maximum of thirty four⁴ images. In each image, for the same scene point, the effective distance from the image plane to lens is different. There is a 1-to-1 relationship between the image column number and the distance from lens to image, and therefore, by the lens law, a 1-to-1 relationship between the image column number and the range of the scene point. The column number at which a scene point is imaged with greatest sharpness, is therefore also a measure of the range.

Results Among the focus criterion functions that were tried, the Tennegrad function [17] seemed to have the best performance/speed characteristics. In addition to problems like depth of field, lack of detail, selection of window size etc., that are present in most range-from-focus algorithms, the range map has two problems as described below.

- Consider a scene point, A, that is imaged on pixels, $I_1[230][470]$, $I_2[230][455]$, $I_3[230][440]$... Consider also a neighboring scene point B, that is imaged on pixels $I_1[230][471]$, $I_2[230][456]$, $I_3[230][441]$... The focus criterion values for point A will peak at a column number that is $470 - n \times 15$ (where $0 \leq n$). If point B is also at the same range as A, then the focus criterion values for point B will peak at a column number that is $471 - n \times 15$, for the same n as that for point A. The peak column number for point A will therefore be 1 less than that of point B. If we have a patch of points that are all at the same distance from the camera, then the peak column numbers obtained will be numbers that change by 1 for neighboring points⁵. The resulting range map therefore shows a local ramping behavior.
- As we mentioned before, a scene point is imaged about 34 times, at different levels of sharpness (or blur). It is very likely that the least blurred image would have been obtained for some camera



(a)



(b)

Figure 4: (a) Panning camera, environment fixed range array, and the images obtained at successive pan angles. Each range array element is associated with multiple criterion function values which are computed from different overlapping views. The maximum of the values in any radial direction is the one finally selected for the corresponding range array element, to compute the depth value in that direction. (b) Steps involved in obtaining range from focus.

⁴Roughly $\frac{512}{15}$

⁵Neighbours along vertical columns will not have this problem

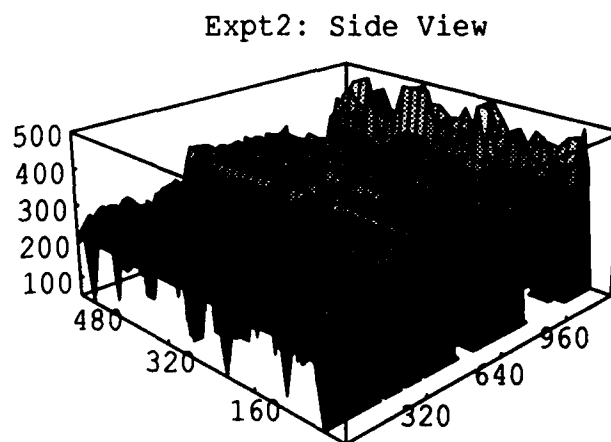
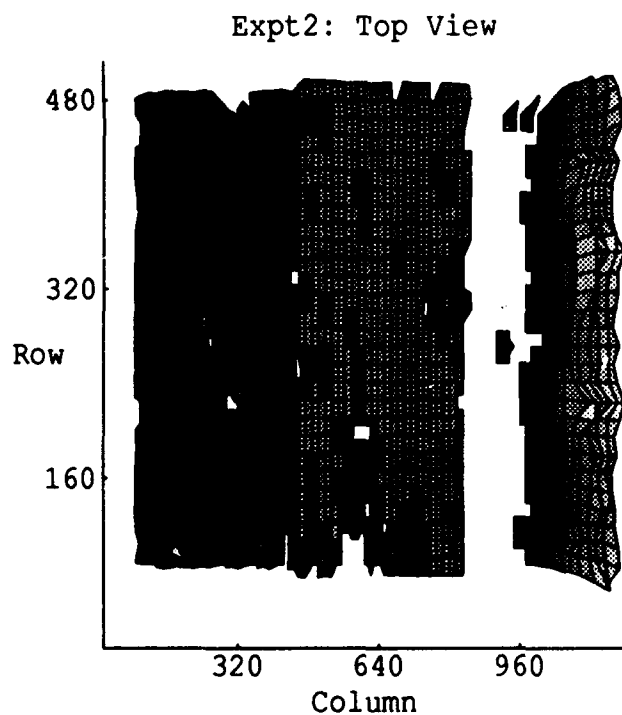
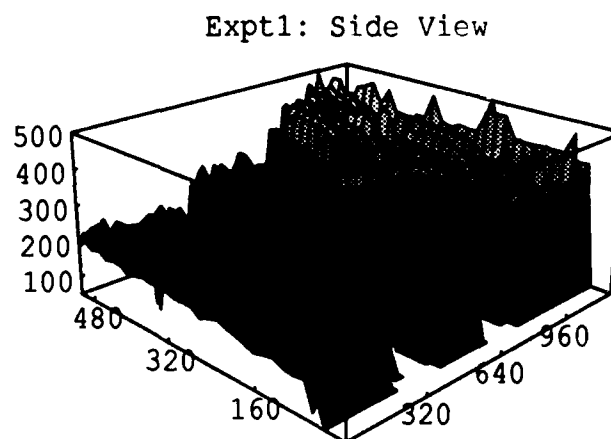
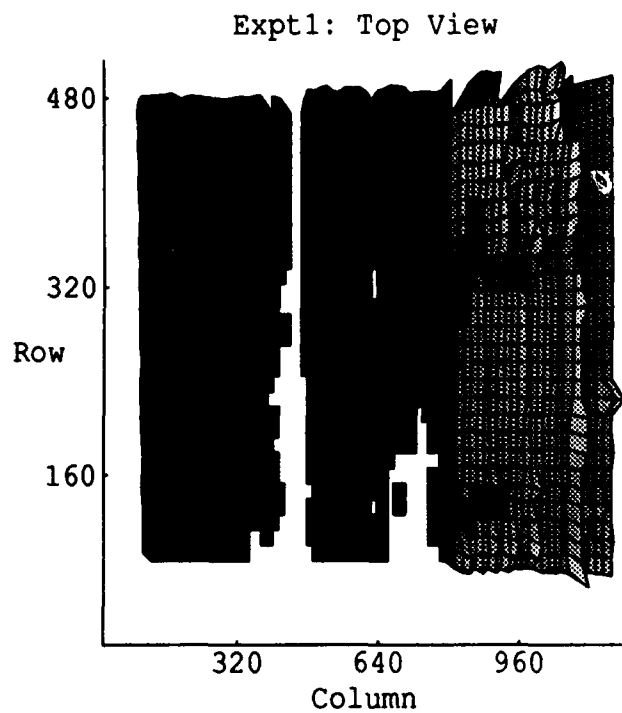


Figure 5: Range disparities for experiments 1 and 2. Parts of the scene for which range values could not be calculated are shown blank. The further away a surface is from the camera, the smaller is its height in the range disparity map.

parameter that corresponds to a value between two input frames.

To reduce these problems, we fit a gaussian to the three focus criterion values around the peak to determine the location of the real maximum. For brevity, we have not included some sample images from the experiments. Figure 5 shows two views of the range disparity values for experiments 1 and 2. Parts of the scene where we cannot determine the range disparity values are shown blank.

6 SUMMARY AND CONCLUSIONS

In this paper we have shown that using a camera whose image plane is not perpendicular to the optical axis, allows us to determine estimates of range values of object points. We showed that the SF surface, which appears in sharp focus when imaged by our non-frontal imaging camera, is an inclined plane. When the camera's pan angle direction changes, by turning about the lens center, an SF volume is swept out by the SF surface. The points within this volume comprise those for which range can be estimated correctly. We have described an algorithm that determines the range of scene points that lie within the SF volume. We point out some of the shortcomings that are unique to our method. We have also described the results of some experiments that were conducted to prove the feasibility of our method.

References

- [1] A. Lynn Abbott and Narendra Ahuja. Active surface reconstruction by integrating focus, vergence, stereo, and camera calibration. In *Proc. Third Intl. Conf. Computer Vision*, pages 489-492, Osaka, Japan, Dec. 1990.
- [2] T. Darrell and K. Wohn. Pyramid based depth from focus. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 504-509, 1988.
- [3] Subhdev Das and Narendra Ahuja. Multiresolution image acquisition and surface reconstruction. In *Proc. Third Intl. Conf. Computer Vision*, pages 485-488, Osaka, Japan, Dec. 1990.
- [4] Subhdev Das and Narendra Ahuja. Active surface estimation: Integrating coarse-to-fine image acquisition and estimation from multiple cues. Technical Report CV-92-5-2, Beckman Institute, University of Illinois, 1992.
- [5] Subhdev Das and Narendra Ahuja. Performance analysis of stereo, vergence, and focus as depth cues for active vision. Technical Report CV-92-6-1, Beckman Institute, University of Illinois, 1992.
- [6] John Ens and Peter Lawrence. A matrix based method for determining depth from focus. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 600-606, Maui, Hawaii, June 1991.
- [7] Berthold Klaus Paul Horn. Focusing. Technical Report 160, MIT Artificial Intelligence Lab, Cambridge, Mass., 1968.
- [8] R. A. Jarvis. Focus optimisation criteria for computer image processing. *Microscope*, 24:163-180, 1976.
- [9] E. P. Krotkov. Focusing. Technical Report MS-CIS-86-22, GRASP Laboratory, University of Pennsylvania, April 1986.
- [10] E. P. Krotkov, J. Summers, and F. Fuma. Computing range with an active camera system. In *Eighth International Conference on Pattern Recognition*, pages 1156-1158, October 1986.
- [11] Eric P. Krotkov. *Active Computer Vision by Cooperative Focus and Stereo*. New York: Springer-Verlag, 1989.
- [12] G. Ligthart and F. C. A. Groen. A comparison of different autofocus algorithms. In *Proc. Sixth Intl. Conf. Pattern Recognition*, pages 597-600, Oct. 1982.
- [13] Shree K. Nayar and Yasuo Nakagawa. Shape from focus: An effective approach for rough surfaces. In *Proc. IEEE Intl. Conf. Robotics and Automation*, pages 218-225, Cincinnati, Ohio, May 1990.
- [14] Alex Paul Pentland. A new sense for depth of field. *IEEE Trans. Pattern Anal. and Machine Intell.*, PAMI-9:523-531, 1987.
- [15] J. F. Schlag, A. C. Sanderson, C. P. Neuman, and F. C. Wimberly. Implementation of automatic focusing algorithms for a computer vision system with camera control. Technical Report CMU-RI-TR-83-14, Carnegie-Mellon University, August 1985.
- [16] G. Sperling. Binocular vision: A physical and a neural theory. *Amer. J. Psychology*, 83:461-534, 1970.
- [17] Jay Martin Tenebaum. *Accommodation in Computer Vision*. PhD thesis, Stanford University, Palo Alto, Calif., 1971.

Depth from Focusing and Defocusing

Yalin Xiong
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
email: yx@cs.cmu.edu

Steve Shafer
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
email: sas@cs.cmu.edu

Abstract

This paper studies the problem of obtaining depth information from focusing and defocusing, which have long been noticed as important sources of depth information for human and machine vision. In depth from focusing, we try to eliminate the local maxima problem which is the main source of inaccuracy in focusing; in depth from defocusing, a new computational model is proposed to achieve higher accuracy.

The major contributions of this paper are: (1) In depth from focusing, instead of the popular Fibonacci search which is often trapped in local maxima, we propose the combination of Fibonacci search and curve fitting, which leads to an unprecedentedly accurate result; (2) New model of the blurring effect which takes the geometric blurring as well as the imaging blurring into consideration, and the calibration of the blurring model; (3) In spectrogram-based depth from defocusing, a maximal resemblance estimation method is proposed to decrease or eliminate the window effect.

This paper reports focus ranging with less than 1/1000 error and the defocus ranging with less than 1/200 error. With this precision, depth from focus ranging is becoming competitive with stereo vision for reconstructing 3D depth information.

1 Introduction

Obtaining depth information by actively controlling camera parameters is becoming more and more important in machine vision, because it is passive and monocular. Compared with the popular stereo method for depth recovery, this focus method doesn't have the correspondence problem, therefore it is a valuable method as an alternative of the stereo method for depth recovery.

There are two distinct scenarios for using focus in-

formation for depth recovery:

- *Depth From Focus*: We try to determine distance to one point by taking many images in better and better focus. Also called "autofocus" or "software focus". Best reported result is 1/200 depth error at about 1 meter distance [Subbarao, 1992].
- *Depth From Defocus*: By taking small number of images under different lens parameters, we can determine depth at all points in the scene. This is a possible range image sensor, competing with laser range scanner or stereo vision. Best reported result is 1.3% RMS error in terms of distance from the camera when the target is about 0.9 m away [Ens and Lawrence, 1991].

Both methods have been limited in past by low precision hardware and imprecise mathematical models. In this paper, we will improve both:

- *Depth From Focus*: We propose a stronger search algorithm with its implementation on a high precision camera motor system.
- *Depth From Defocus*: We propose a new estimation method and a more realistic calibration model for the blurring effect.

With this new results, focus is becoming viable as technique for machine vision applications such as terrain mapping and object recognition.

2 Depth From Focusing

Focusing has long been considered as one of major depth sources for human and machine vision. In this section, we will concentrate on the precision problem of focusing. We will approach high precision from both software and hardware directions, namely, stronger algorithms and more precise camera system.

Most previous research on depth from focusing concentrated on developments and evaluations of different focus measures, such as [Tenenbaum, 1970,

Krotkov, 1987, Nayar and Nakagawa, 1990, Subbarao *et al.*, 1992]. As described by all these researchers, an ideal focus measure should be unimodal, monotonic, and should reach the maximum only when the image is focused. But the focus measure profile has many local maxima due to noises and/or the side-lobe effect ([Subbarao *et al.*, 1992]) even after magnification compensation ([Willson and Shafer, 1991]). This essentially requires a more complicated peak detection method compared with the Fibonacci search which is optimal under the unimodal assumption as in [Beveridge and Schechter, 1970, Krotkov, 1987]. In this paper, we use a recognized focus measure from the literature, which is the Tenegrad with zero threshold in [Krotkov, 1987] or M_2 method in [Subbarao *et al.*, 1992]. Our major concern is to discover to what extent the precision of focus ranging can scale up with more precise camera systems and more sophisticated search algorithms. We propose the combination of Fibonacci search and curve fitting to detect the peak of focus measure profile precisely and quickly.

To evaluate the results from peak detections, an error analysis method is presented to analyze the uncertainty of the peak detection in the motor count space, and to convert the uncertainty in the motor count space into uncertainty of depth. We compute the variance of motor positions resulted from peak detections over equal depth targets. The Rayleigh criterion of resolution is applied to the distribution of motor positions to calculate the minimal differentiable motor displacement. With the assumption of local linearity of the mapping from the motor count space to focus depth, the minimal differentiable motor displacement can be converted to the minimal differentiable depth.

The lack of high precision equipment has been a limiting factor to previous implementations of various focus ranging methods. Many implemented systems, such as SPARCS, have fairly low motor resolution, which actually prohibits more precise results. We will give brief description of the motor-driven camera system in Calibrated Imaging Lab later, and further details can be found in [Willson and Shafer, 1992].

2.1 Fibonacci Search and Curve Fitting

When the focus motor resolution is high, we usually have a very large parameter space which prevents us from exhaustively searching all motor positions. Based on the unimodal assumption of focus measure profile, Fibonacci search was employed to narrow the parameter space down to the peak [Beveridge and Schechter, 1970].

Assume the initial interval is $[x, y]$, and we know the focus measure profile is unimodal in this interval, if $x < x_1 < x_2 < y$ and $F(x_1) < F(x_2)$, where F is the focus measure function, then the peak can not

be within interval $[x, x_1]$, otherwise the unimodal assumption will be violated. Therefore, if we can properly choose x_1 and x_2 , the peak can be found optimally. Fibonacci search is the optimal search under the unimodal assumption.

Figure 1 shows the target used for testing the focus measure, and Figure 2 is the focus measure profile of the target.

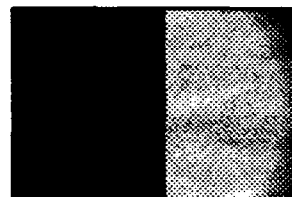


Figure 1: Step Edge Image as Target

It is clear from Figure 2 that Fibonacci search will fail to detect the peak precisely because of the jagged profile. Fortunately, those local maxima are small in size, and therefore can be regarded as disturbances. From previous paragraphs, we know that the Fibonacci search only evaluates at two points within the interval, which gives rise to the hope that when the interval is large, Fibonacci search is still applicable because it will overlook those small ripples.

As the search goes on, the interval becomes smaller and smaller. Consequently, Fibonacci search must be aborted at some point when the search might be misleading. We can experimentally set up a threshold, when the length of the interval is less than the threshold, Fibonacci search is replaced by an exhaustive search. After the exhaustive search, a curve is fitted to the part of profile resulting from the exhaustive search.

In our experiments, we set the threshold to be 5 motor counts. So when the Fibonacci search narrows down the whole motor space to $[a, b]$, where $b - a \leq 5$, an exhaustive search is fired on the interval $[a - c, b + c]$, where c is a positive constant. A Gaussian function is fitted to the profile in the interval $[a - c, b + c]$ using the least square method described in [Press *et al.*, 1988].

Figure 3 shows the result when Fibonacci search alone is applied to the focus measure profile. Apparently, the search is trapped in a local maximum. Figure 4 shows the result from Gaussian function fitting. Both graphs show only a part of the whole motor space.

2.2 Error Analysis

Because of the depth accuracy we expected, a direct measurement of absolute depth is impossible. Instead, we prefer to use the minimal differentiable depth as an indication of the depth accuracy. If we assume the peak motor positions resulting from

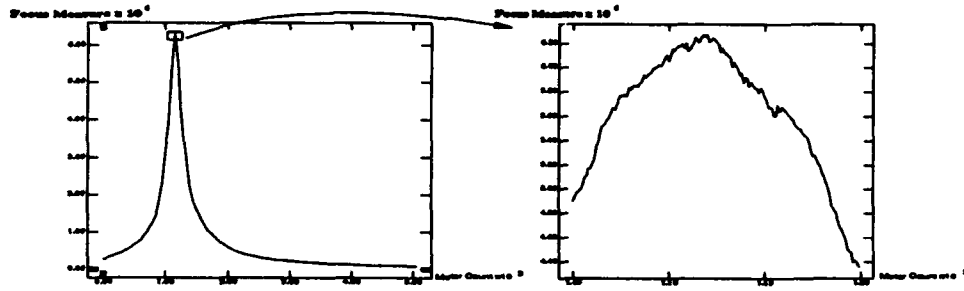


Figure 2: Focus Measure Profile

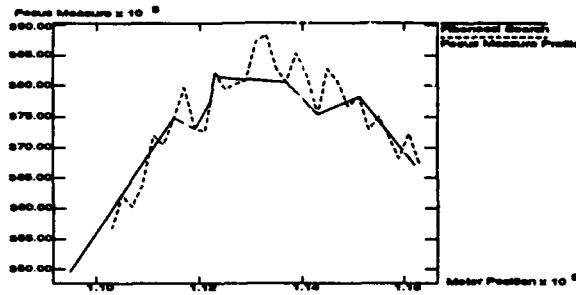


Figure 3: Fibonacci Search

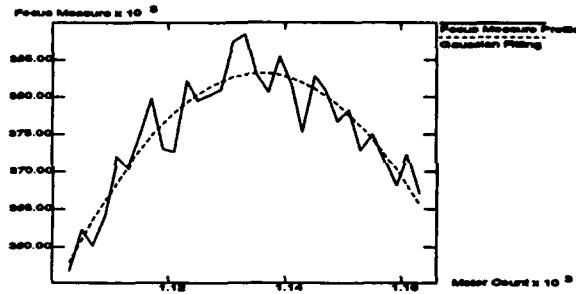


Figure 4: Curve Fitting

the same repeated experiments have a Gaussian distribution, we can define the minimal differentiable motor displacement as the minimal difference of two motor counts which have pre-defined probability of representing different peaks. For example, in Figure 5, the Gaussian distribution is artificially cut at 90% line, so we can say, if we do one focus ranging experiment on a target at the depth corresponding to the peak a , there is a 90% of probability the motor count will be within the interval A .

There can be different pre-defined probability for the definition of minimal differentiable motor displacement. We define the minimal differentiable motor displacement based on Rayleigh criterion for resolution [Born and Wolf, 1964] which specifies the saddle-to-peak ratio as $8/\pi^2$. In case of the Gaussian distribution, the cut-off line corresponding to the Rayleigh criterion is about 0.9σ .

There is a mapping from a motor count to an absolute depth value definitely. Assume $d = f(m)$ where

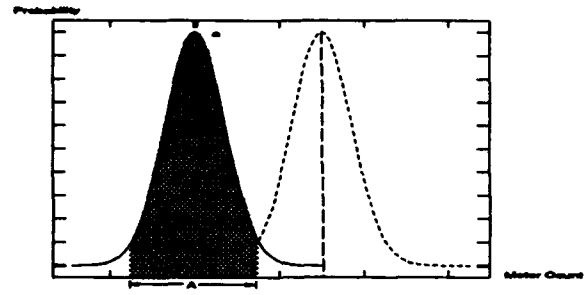


Figure 5: Minimal Differentiable Motor Displacement

d is the depth, m the motor count and f the mapping, we have

$$\frac{\Delta d}{\Delta m} = f'(m), \quad (1)$$

where $f'(m)$ is the first order derivative with respect to m . Because what we really want to know is the minimal differential depth or depth resolution Δd , and we already have the minimal differentiable motor displacement Δm , the only thing need to be calibrated is $f'(m)$. If we assume $f'(m)$ is a constant in the vicinity of $d = D$, and the motor count distribution has its center at $m = M$, then when the target is moved ΔD , the distribution center moves ΔM , and we will have the minimal differentiable depth

$$\Delta d = \frac{\Delta m}{\Delta M} \Delta D \quad (2)$$

where Δm is the minimal differentiable motor displacement.

2.3 Implementation and Result

2.3.1 Hardware

We implemented this focus ranging algorithm in the Calibrated Imaging Laboratory, using the Fujinon/Photometric camera system [Willson and Shafer, 1992]. The focal length can change between 10 mm to 130 mm with 11100 motor steps, and the focus distance can change from approximately 1 meter to infinity with 5100 motor steps, the aperture can change from F1.7 to completely closed with 2700 motor steps. The SNR of the camera can be as low

as 400/1 because of the pixel by pixel digitization and the -40°C temperature of the sensor.

2.3.2 Experiments and Results

We put the target of Figure 1 at about 1.2 meters away from the front lens element of the camera. Maximal focal length and maximal aperture are employed to achieve the minimal depth of field. The evaluation window is 40x40, while the gradient operator is a 3x3 Sobel operator.

The distribution of motor positions are sketched in Figure 6 resulting from an experiment repeated 40 times. With the mean as the center of a Gaussian, and the standard deviation as σ of the Gaussian, we have the minimal differentiable motor displacement as $2 \times 0.9 \sigma = 4.5$ motor counts.



Figure 6: Motor Position Distribution

Then the target is moved toward the camera 1 centimeter, and we repeated the above experiments. The center of the motor count distribution moves 38 counts. Therefore, by Eq. 2, we have the minimal differentiable depth:

$$\Delta d = \frac{\Delta m}{\Delta M} \Delta D = \frac{4.5}{38} \times 1\text{cm} = 0.118\text{cm}. \quad (3)$$

And the relative depth error is about $0.118 / 120 = 0.098\%$.

3 Depth From Defocusing

The depth from defocusing method uses the direct relationships among the depth, camera parameters and the amount of blurring in images to derive the depth from parameters which can be directly measured. In this part of the paper, we propose the maximal resemblance estimation method to estimate the amount of defocusing accurately, and a calibration-based blurring model.

Because the blurring in an image can be caused by either the imaging process or the scene itself, it generally requires at least two images taken under different camera configurations to eliminate this ambiguity. Pentland solved this problem by taking one picture by a pin-hole camera, which can be regarded as the orthographic projection of the scene with zero

imaging blurring, and another one by a wide aperture camera [Pentland, 1987]. In this paper, we intend to employ two images which are defocused to different extents.

Window effects have largely been ignored in the literature of this field, except [Ens and Lawrence, 1991, Ens, 1990], where the author derived a function of RMS depth error in terms of the size of window. For example, when the window is 4 pixels by 4 pixels, the RMS error from the window effect can be as large as 65.8%! The maximal resemblance estimation method we propose is capable of eliminating the window effect. It is also noticed that the size of the window is the decisive factor that limits the resolution of depth maps if we try to obtain a dense depth map. Therefore if we can use smaller window without reducing the quality of the results, the resolution of dense depth maps can be much higher.

Previous work has employed oversimplified camera models to derive the relationship between blurring functions and camera configurations. In [Pentland, 1987, Subbarao, 1988, Bove, 1989], the radius of blurring circles are derived from the ideal thin lens model. In this paper, we will propose a more sophisticated function which directly relates the blurring function with camera motors. Experimental results are very consistent with this model as to be shown later.

3.1 Background

The depth from defocus method is based on the idea that, the amount of blurring change is directly related to the depth and camera parameters. Since the camera parameters can be calibrated, the depth can be expressed by the amount of blurring change correspondingly. To estimate the amount of blurring change, we need a model of the optical blurring. Traditionally, the blurring effect is modeled as the convolution of Gaussian in computer vision literature, partly due to its mathematical tractability. Here we will still assume a Gaussian model, i.e. (considering 1D case):

$$I(x) = \int_{-\infty}^{+\infty} I_0(\xi) g_{\sigma(d,c)}(x - \xi) d\xi \quad (4)$$

$$g_{\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \quad (5)$$

The basic idea of the depth-from-defocus method is that, in Eq. 4, since the $I(x)$ which is the image and c which results from camera calibration are known, and d and $I_0(x)$ are unknown, we can take two images under different camera settings c_1 and c_2 , then at least theoretically d can be computed. But because the Eq. 4 is not a linear equation with respect to unknown d , directly solving d is either impossible or numerically unstable. Pentland proposed a

method to solve d by Fourier transforms [Pentland, 1987]:

If

$$I_1(x) = I_0(x) * g_{\sigma_1}(x) \quad (6)$$

$$I_2(x) = I_0(x) * g_{\sigma_2}(x) \quad (7)$$

then,

$$\begin{aligned} \ln \frac{\mathcal{F}[I_1(x)]}{\mathcal{F}[I_2(x)]} &= \ln \frac{\mathcal{I}_1(f)}{\mathcal{I}_2(f)} = \ln \frac{\mathcal{I}_0(f)\mathcal{G}_{\sigma_1}(f)}{\mathcal{I}_0(f)\mathcal{G}_{\sigma_2}(f)} \\ &= \ln \frac{\mathcal{G}_{\sigma_1}(f)}{\mathcal{G}_{\sigma_2}(f)} \\ &= -\frac{1}{2}f^2(\sigma_1^2 - \sigma_2^2) \end{aligned} \quad (8)$$

$$\sigma_1 = \sigma(d, c_1) \quad (9)$$

$$\sigma_2 = \sigma(d, c_2) \quad (10)$$

Replacing Eq. 9 and Eq. 10 into Eq. 8, we can get:

$$\ln \frac{\mathcal{I}_1(f)}{\mathcal{I}_2(f)} = -\frac{1}{2}f^2(\sigma^2(d, c_1) - \sigma^2(d, c_2)) \quad (11)$$

where the function σ can be calibrated.

Obviously, in Eq. 11, the only unknown is d , therefore, depth recovery from two images is straightforward.

3.2 Gabor Transform and Window Effect

The method explained above is based on $\mathcal{F}[I(x)]$, which is the Fourier transform of the entire image. Thus, only one d can be calculated from the entire image. If our goal is to obtain a dense depth map $d(x, y)$, we are forced to use the STFT (Short Time Fourier Transform) to preserve the depth locality. To eliminate the spurious high frequency components generated by the discontinuity at the window boundary, people usually multiply the window by a window function. Unfortunately, the elegant cancellation in Eq. 8 doesn't hold any more if we introduce the window function $W(x)$:

$$\begin{aligned} \ln \frac{\mathcal{F}[I_1(x)W(x)]}{\mathcal{F}[I_2(x)W(x)]} &= \ln \frac{\mathcal{I}_1(f) * \mathcal{W}(f)}{\mathcal{I}_2(f) * \mathcal{W}(f)} \\ &= \ln \frac{(\mathcal{I}_0(f)\mathcal{G}_{\sigma_1}(f)) * \mathcal{W}(f)}{(\mathcal{I}_0(f)\mathcal{G}_{\sigma_2}(f)) * \mathcal{W}(f)} \end{aligned} \quad (12)$$

The convolutions in Eq. 12 introduce blurring in both the time (space) and frequency domains. The Gabor transform [Rioul and Vetterli, 1991], which uses a Gaussian as the window function, can minimize the production of spatial uncertainty and spectral uncertainty. Assuming a Gaussian function is used as $W(x) = g_{\sigma_w}(x)$, and its Fourier transform is $\mathcal{W}(f) = e^{-\sigma_w^2 f^2/2}$, we have:

$$\Delta f^2 = \frac{\int f^2 |\mathcal{W}(f)|^2 df}{\int |\mathcal{W}(f)|^2 df} = \frac{1}{2\sigma_w^2} \quad (13)$$

$$\Delta x^2 = \frac{\int x^2 |W(x)|^2 dx}{\int |W(x)|^2 dx} = \frac{\sigma_w^2}{2} \quad (14)$$

The above equations state that, in the frequency domain, two frequency components Δf away can't be discriminated, and in the space domain, two impulses Δx away can't be discriminated either. Apparently, one interpretation of why Eq. 8 doesn't hold is that Δf is not zero.

3.3 Maximal Resemblance Estimation

From observation, we know that when σ_1 is approaching σ_2 , Eq. 12 also approaches zero, in other words, when σ_1 almost equals σ_2 , the Eq. 8 can be a good approximation in terms of absolute error. This observation suggests an iterative method in which the blurring difference Δ is refined by blurring one image to resemble the other in the vicinity of one pixel. In symbols: (Assuming $\Delta_{(k)}$ is the the k th estimation of $\sigma_1^2 - \sigma_2^2$)

1. $I_1^{(0)} = I_1, I_2^{(0)} = I_2$ and $\Delta = 0.0, k = 0$;
2. $\mathcal{I}_1^{(k)} = \mathcal{F}[I_1^{(k)}W]$,
 $\mathcal{I}_2^{(k)} = \mathcal{F}[I_2^{(k)}W]$.
3. Fit a curve to $\ln \frac{\mathcal{I}_1^{(k)}}{\mathcal{I}_2^{(k)}} = -f^2 \Delta_{(k)}/2$. (Refer to Eq. 8)
4. $\Delta = \sum_{i=0}^k \Delta_{(i)}$.
5. If $\Delta > 0$, then
 $I_1^{(k+1)} = I_1$;
 $I_2^{(k+1)} = I_2 * G_{\sigma=\sqrt{\Delta}}$;
else,
 $I_1^{(k+1)} = I_1 * G_{\sigma=\sqrt{-\Delta}}$;
 $I_2^{(k+1)} = I_2$;
- Note all these convolutions are done very locally because of the window function multiplication in step 2.
6. If the termination criteria are satisfied, exit.
7. $k = k+1$, go to step 2.

All above operations involve only local pixels, and don't require taking new pictures. Therefore, the computation can be done in parallel to all pixels to obtain a dense depth map.

Let's trace the above iterations, at the first cycle, we have (Assuming $\sigma_1 > \sigma_2$):

$$\Delta_{(0)} = (\sigma_1^2 - \sigma_2^2) + E_{(0)}$$

while $E_{(0)}$ is the error of estimating $\sigma_1^2 - \sigma_2^2$.

$$I_1^{(1)} = I_1 = I_0 * G_{\sigma_1}$$

$$I_2^{(1)} = I_2 * G_{\sqrt{\Delta}} = I_0 * G_{\sqrt{\sigma_1^2 + E_{(0)}}}$$

We can see, after the first iteration, we actually switched to estimate $E_{(0)}$. So after k iterations, we have

$$\Delta = \sum_{i=0}^k \Delta_{(i)} = \sigma_1^2 - \sigma_2^2 + E_{(k)} \quad (15)$$

Now the problem is whether the sequence $E_{(k)}$ ($k = 0, 1, 2, \dots$) converges to zero. Unfortunately, there is no way to prove this mathematically because it depends on the fitting method used in step 3. Notice that if in step 3, we get an estimate of $\sigma_1^2 - \sigma_2^2$ only based on one particular frequency component, $E_{(k)}$ may diverge. Previous depth from defocus methods usually counted on a pre-selected frequency band, such as in [Pentland, 1987], sometimes this may cause a very large error if there is not enough energy of the image content within that frequency band.

3.4 Fitting Algorithm

Common to any frequency analysis, we need a robust algorithm to extract $\sigma_1^2 - \sigma_2^2$ in Eq. 8 in a noisy environment. Ignoring the phase information resulting from Gabor transform, Eq. 8 becomes:

$$\ln \frac{|I_1(f)|^2}{|I_2(f)|^2} = -f^2(\sigma_1^2 - \sigma_2^2) \quad (16)$$

Assuming an additive white noise model, we have:

$$\ln \frac{|I_1(f)|^2 + n_1}{|I_2(f)|^2 + n_2} = \ln(|I_1(f)|^2 + n_1) - \ln(|I_2(f)|^2 + n_2), \quad (17)$$

where n_1 and n_2 are energy of noises. Because $\ln(x + dx) \approx \ln x + \frac{1}{x}dx$, if we assume $|I_1(f)|^2 \gg n_1$ and $|I_2(f)|^2 \gg n_2$, Eq. 17 can be approximated as:

$$\ln \frac{|I_1(f)|^2}{|I_2(f)|^2} + \left(\frac{n_1}{|I_1(f)|^2} - \frac{n_2}{|I_2(f)|^2} \right) \quad (18)$$

Therefore, at each frequency, the left hand of Eq. 16 can be approximated by dividing corresponding spectral energy of two images at the specific frequency, provided that the energy in that frequency is much larger than the energy of noise. The deviation of this approximation can be expressed as:

$$\sigma_f = c_n \left(\frac{1}{|I_1(f)|^2} + \frac{1}{|I_2(f)|^2} \right) \quad (19)$$

where c_n is a constant related to the noise energy of the camera.

Certainly, Eq. 19 is an approximation to model the error distribution as an Gaussian. As an intuition, when $|I_1(f)|^2$ or $|I_2(f)|^2$ is large, i.e. the energy within the frequency is high, the deviation is small, and *vice versa*.

From Eq. 16, we know this estimation problem is a typical linear regression problem. With the uncertainty measurement approximated in Eq. 19, $\sigma_1^2 - \sigma_2^2$ can be estimated robustly. More details about linear regression methods can be found in [Press *et al.*, 1988].

There is one more problem need to be addressed. Since we obtain images under different camera configurations, the total energy within one image is different from that within the other. Usually, a brightness normalization is performed to every image before Gabor transforms, as in [Subbarao and Wei, 1992]. But since this normalization will have different effects over the noise in two images, which will complicate the uncertainty analysis, we prefer not to normalize the brightness in two images. Instead we assume:

$$I_1(x) = c_1 I_0(x) * g_{\sigma_1}(x) \quad (20)$$

$$I_2(x) = c_2 I_0(x) * g_{\sigma_2}(x), \quad (21)$$

where c_1 and c_2 are two unknown constants. Replacing the two equations into Eq. 16, we have:

$$\ln \frac{|I_1(f)|^2}{|I_2(f)|^2} = -f^2(\sigma_1^2 - \sigma_2^2) + 2 \ln \frac{c_2}{c_1}, \quad (22)$$

which is still a linear problem, while the uncertainty analysis still holds.

3.5 Blurring Model

Since the defocus ranging method derives the depth instead of searching for the depth, it requires a direct modeling of defocusing in terms of camera parameters and depth. Previous researchers usually derived the relation among lens parameters, the depth and the blurring radius, such as in [Pentland, 1987, Subbarao, 1988]. For example, in [Pentland, 1987], by simple geometric optics, Pentland derived the formula:

$$D = \frac{F v_0}{v_0 - F - \sigma k f} \quad (23)$$

where D is the depth, F the focal length, f the f -number of the lens, v_0 the distance between lens and image plane, σ the blurring circle radius, and k a constant.

The basic limitation of this approach is that those parameters are based on the ideal thin lens model and in fact, they can never be measured precisely on any camera. We desire a function which is in terms of motor counts, which are measurable and controllable. For instance, if we use m_z for zoom motor count, m_f for focus motor count, and m_a for aperture motor count, we wish to get a function in the form of $D = F(m_z, m_f, m_a, \sigma)$ or $\sigma = F(m_z, m_f, m_a, D)$. Due to the depth ambiguity in the former form ([Pentland, 1987], Appendix), we prefer to express the blurring radius σ as a function of motor counts and the depth.

From Eq. 23, we can express σ as:

$$\sigma = \frac{v_0 - F}{kf} - \frac{Fv_0/kf}{D} \quad (24)$$

Since all the lens parameters can be thought as derived from motor counts, we can rewrite Eq. 24 as:

$$\sigma = k_1(m_s, m_f, m_a) + \frac{k_2(m_s, m_f, m_a)}{D + k_3(m_s, m_f, m_a)} \quad (25)$$

Notice that there is another term k_3 added. Because D is the distance between lens and object, and the position of lens changes as camera parameters are changed, we intend to use a fixed plane perpendicular to the optical axis as the depth reference plane at $z = k_3$. From now on, we always refer to depth as the distance between the target and the depth reference plane.

Eq. 25 says, at some point, σ will drop to zero, which is the best focused point. But we knew that we can never get a real step edge, because there is always high frequency loss in the imaging process. It can be attributed to the pixel quantization, diffraction, etc. Similarly, we can model this as a convolution with a Gaussian independent of the geometric blurring which we already modeled. We use k_4 to model its width.

Since two consecutive convolutions with Gaussians are equivalent with one Gaussian convolution:

$$G_{\sigma_1} * G_{\sigma_2} = G_{\sqrt{\sigma_1^2 + \sigma_2^2}}, \quad (26)$$

we have our final blurring model expressed as:

$$\sigma^2 = \left(k_1(m_s, m_f, m_a) + \frac{k_2(m_s, m_f, m_a)}{D + k_3(m_s, m_f, m_a)} \right)^2 + k_4^2(m_s, m_f, m_a) \quad (27)$$

3.6 Implementation and Results

3.6.1 Simulation

Our first simulation examines how precise the estimate of $\sigma_1^2 - \sigma_2^2$ can be. We use step function as I_0 , and convolve it with two different Gaussians G_{σ_1} and G_{σ_2} . The window function is also a Gaussian with σ equals to three pixel widths.¹ From Eq. 14, the locality of the window function is about 2 pixel widths.

The result of the iterative method is illustrated in Fig. 7. And we can see that, when the window function is narrow, how poor the first estimation can be. As the iteration goes on, the estimated value converges fast to the true value. Experimentally, the final error is lower bounded by the discretization of

¹In this paper, all σ values are in pixel width.

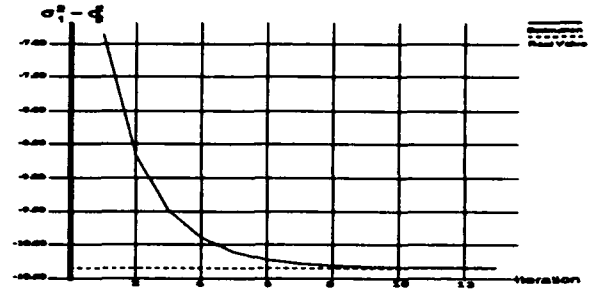


Figure 7: Iterative Estimation of $\sigma_1^2 - \sigma_2^2$

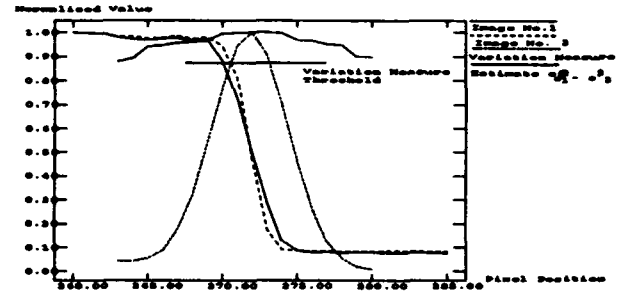


Figure 8: Variation Measure and Threshold

the functions, that is, when the σ of the Gaussian function is too small with respect to the pixel width, the discrete Gaussian is no longer a good approximation of the real Gaussian function, and the results begin to degenerate. Generally, in the absence of noise, the estimation errors are less than 1/1000 of the true values.

3.6.2 Variation Measure and Thresholding

Certainly, if there is no texture or little texture within the image, we can not expect to obtain accurate estimation of depth. Therefore, we need a measure which can discriminate image patches with enough texture from those without enough texture. Another reason why we need a variation measure is because of the so-called edge bleeding [Nair and Stewart, 1992].

Assuming the image patch is $f(x, y)$, we have the variation measure expressed as in Eq. 28.

$$V = \int \int_{g_{\sigma_1}(x, y)} \left(\left| \frac{\partial f(x, y)}{\partial x} \right|^2 + \left| \frac{\partial f(x, y)}{\partial y} \right|^2 \right) dx dy \quad (28)$$

To better illustrate the relation between the variation measure and the result of estimate of $\sigma_1^2 - \sigma_2^2$, we demonstrate the selection of the threshold to exclude the effects of the low variation content and the edge bleeding in Figure 8.

3.6.3 Calibration of Blurring Function and Blurring Model

First, we tried to confirm our assumption that the blurring function can be approximated by the Gaussian function. Ideally, if we have a point light source, the image of this light source should be the blurring function because

$$\delta(x) * F(x) = F(x). \quad (29)$$

Due to the technical difficulty of an ideal point light source, a step edge image is used instead as shown in Figure 1. Assume the step function is $u(x)$, the image of the step edge should be

$$\begin{aligned} g_\sigma(x) * u(x) &= \int_0^{+\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-t)^2}{2\sigma^2}} dt \\ &= c_1 + c_2 \operatorname{Erf}\left(\frac{x}{\sqrt{2}\sigma}\right) \end{aligned} \quad (30)$$

where Erf is the error function, c_1 and c_2 are two constants.

The Figure 9 illustrates the least square fitting results for a blurred step edge.



Figure 9: Blurring Function Fitting

The coefficients k_1, k_2, k_3, k_4 are constants in Eq. 27 when motors are fixed. We can move the calibration target over four different places, and assume at the first place, the depth of the target is zero (Note the depth is w.r.t. the reference plane), we will have four non-linear equations with four unknowns. To suppress noise, we can measure at more than four places and fit the blurring model to the results.

Using the rail table in CIL ([Willson and Shafer, 1992]), the whole process of calibration can be controlled by the computer. The target moves from about 1 meter from the camera to about 3 meters, and the blurred edges are fed to the least square fitting, the resulting σ 's are, in turn, fitted against the model expressed in Eq. 27.

Experiments have shown very consistent results with the model as illustrated in Figure 10. The target is moved from far to near, at the furthest distance from the camera, the rail motor position is zero. And when it moved through the whole range of the rail, the blurring circle first becomes smaller and smaller,

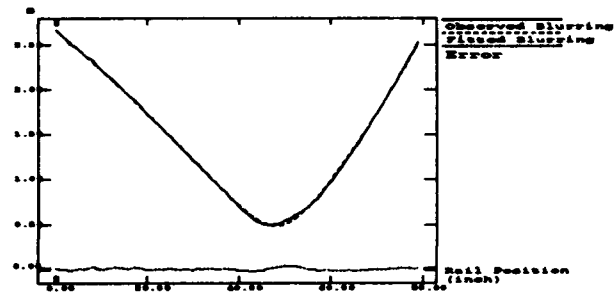


Figure 10: Blurring Model

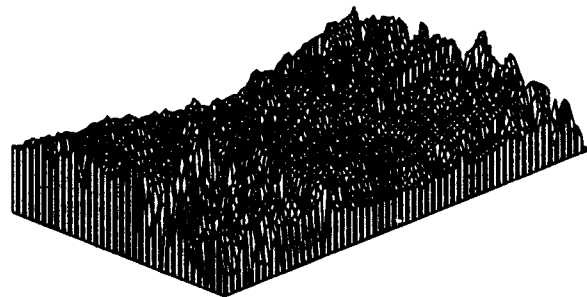


Figure 13: σ^2 -Map Recovery Without Maximal Resemblance Estimation

then after a point, it becomes larger and larger. It is very clear that this effect can be well modeled by Eq. 27.

3.6.4 σ^2 -Map and Shape Recovery

The first step toward a dense depth map is to compute $\sigma^2 = \sigma_1^2 - \sigma_2^2$, without loss of generality we assume $\sigma_1 \geq \sigma_2$, for every pixel, using the maximal resemblance estimation. In Figure 11, we bent a sheet of paper in different directions about 1.0 inches and took images. The target is about 100 inches away from the camera. The focal length is 130mm, the f-number is f/4.7 for (a) and (c), f/8.1 for (b) and (d).

Then we recover σ^2 -map for those two objects. The rectangle in Figure 11 (a) is the area for σ^2 -map. The σ_w for Gabor transform is 5.0 pixel size. Figure 12 shows the σ^2 -map recovery based on the images in Figure 11. The holes within the σ^2 -maps are those patches without enough texture.

Compared with the σ^2 -map recovery without iterative maximal resemblance estimation showed in Figure 13, we can see that results without iteration are much more noisy.

With σ^2 -map recovered and the coefficients in Eq. 27 calibrated w.r.t. the two camera configurations, the depth map recovery is straightforward by using the Brent's method [Press *et al.*, 1988] to numerically solve the nonlinear equation. Figure 14 showed the depth map (in inch) of the convex object in Figure 11 (c) and (d), with respect to the depth refer-

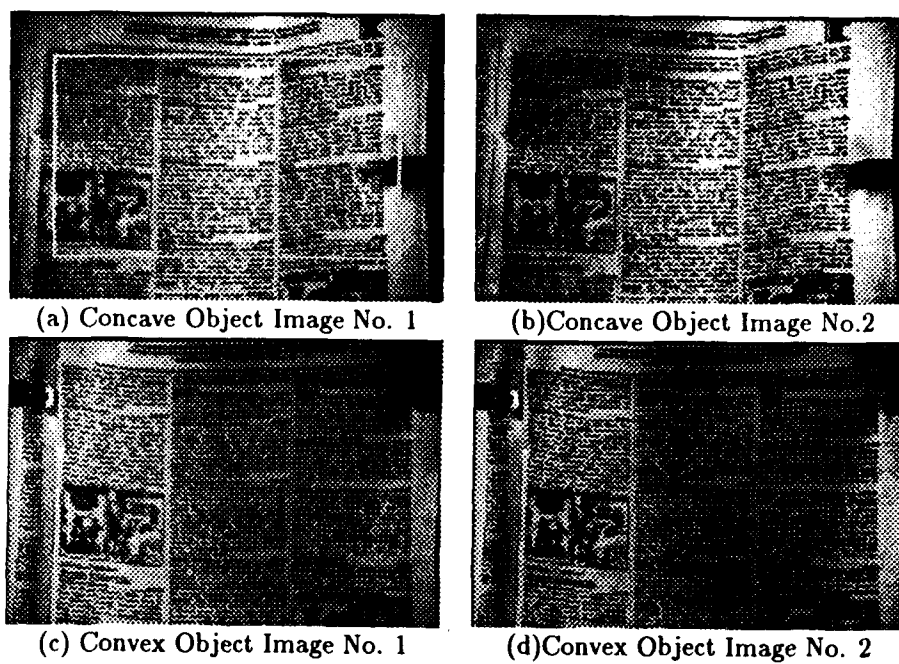


Figure 11: Pictures of Different Objects

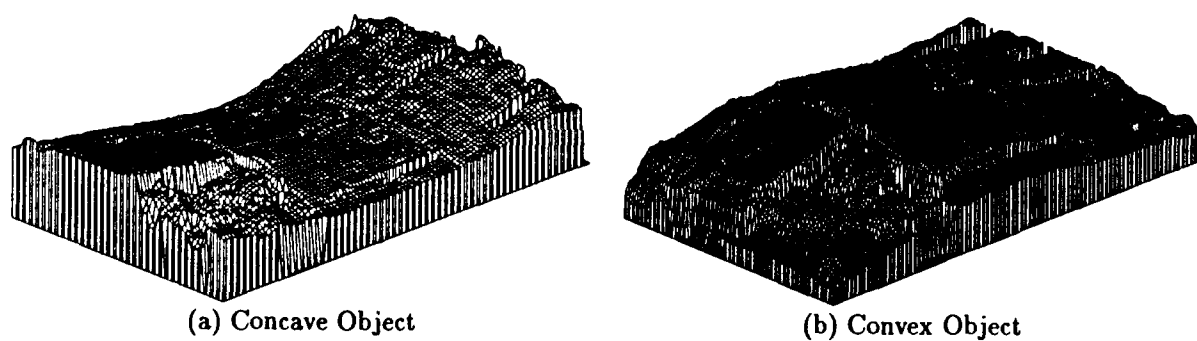


Figure 12: σ^2 -Map Recovery

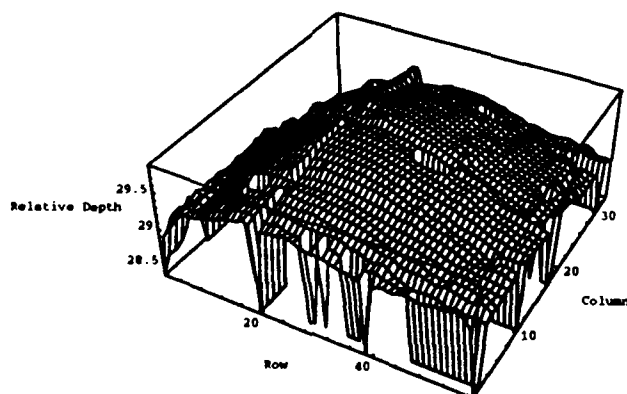


Figure 14: Shape Recovery For the Convex Object

ence plane, which is behind the object. A conservative estimation of depth relative error is 1/200 when the target is 100 inches away.

4 Summary

In summary, we have described two sources of depth information—depth from focusing and depth from defocusing—separately. In depth from focusing, we pursued high accuracy from both the software and hardware directions, and experiments proved that a great improvement was obtained. In depth from defocusing, we re-examined the whole underlying theory, from signal processing to camera calibration, and established a new computational model, which has been successfully demonstrated on real images.

The significance of these works is two-fold. First, there are few previous reports talking about the shape from defocusing or focusing, and the main reason of the inefficacy of shape from defocusing or focusing is its low precision. Demonstrated in this paper, the improvements on precisions of focus ranging and defocus ranging can lead to efficient shape recovery methods. Second, it has been shown that the maximal resemblance method proposed in this paper is capable of preserving the depth locality, which is also essential to obtain dense depth maps.

References

- [Beveridge and Schechter, 1970] G. S. Beveridge and R. S. Schechter. *Optimization: Theory and Practice*. McGraw-Hill, 1970.
- [Born and Wolf, 1964] Max Born and Emil Wolf. *Principles of Optics*. The MACMILLAN COMPANY, 1964.
- [Bove, 1989] V. Michael Bove, Jr. Discrete fourier transform based depth-from-focus. In *Proceedings OSA Topical Meeting on Image Understanding and Machine Vision*, 1989.
- [Darrell and Wohn, 1988] Trevor Darrell and Kwangyeon Wohn. Pyramid based depth from focus. In *Proceedings of CVPR*, pages 504–509, 1988.
- [Ens and Lawrence, 1991] John Ens and Peter Lawrence. A matrix based method for determining depth from focus. In *Proceedings of CVPR*, 1991.
- [Ens, 1990] John Ens. *An Investigation of Methods For Determining Depth From Focus*. PhD thesis, University of British Columbia, 1990.
- [Krotkov, 1987] Eric P. Krotkov. Focusing. *International Journal of Computer Vision*, pages 223–237, 1987.
- [Nair and Stewart, 1992] Hari N. Nair and Charles V. Stewart. Robust focus ranging. In *Proceedings of CVPR*, pages 309–314, 1992.
- [Nayar and Nakagawa, 1990] Shree K. Nayar and Yasuo Nakagawa. Shape from focus: An effective approach for rough surfaces. In *International Conference on Robotics and Automation*, pages 218–225, 1990.
- [Pentland, 1987] Alex P. Pentland. A new sense for depth of field. *IEEE Transactions on PAMI*, 9(4):523–531, 1987.
- [Press et al., 1988] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.
- [Rioul and Vetterli, 1991] Olivier Rioul and Martin Vetterli. Wavelets and signal processing. *IEEE Signal Processing Magazine*, pages 14–38, October 1991.
- [Subbarao and Wei, 1992] Murali Subbarao and Tse-Chung Wei. Depth from defocus and rapid autofocusing: A practical approach. In *Proceedings of CVPR*, pages 773–776, 1992.
- [Subbarao et al., 1992] Murali Subbarao, Tae Choi, and Arman Nikzad. Focusing techniques. Technical Report 92.09.04, Department of Electrical Engineering, State University of New York at Stony Brook, 1992.
- [Subbarao, 1988] Murali Subbarao. Parallel depth recovery by changing camera parameters. In *2nd International Conference on Computer Vision*, pages 149–155, 1988.
- [Subbarao, 1992] Murali Subbarao. Presentation at the symposium on physics-based vision workshop. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1992.
- [Tenenbaum, 1970] J. M. Tenenbaum. *Accommodation in Computer Vision*. PhD thesis, Stanford University, 1970.
- [Willson and Shafer, 1991] Reg G. Willson and Steven A. Shafer. Dynamic lens compensation for active color imaging and constant magnification focusing. Technical Report CMU-RI-TR-91-26, The Robotics Institute, Carnegie Mellon University, 1991.
- [Willson and Shafer, 1992] Reg G. Willson and Steven A. Shafer. Precision imaging and control for machine vision research at Carnegie Mellon University. Technical Report CMU-CS-92-118, School of Computer Science, Carnegie Mellon University, 1992.

A VLSI Smart Sensor for Fast Range Imaging

Andrew Gruss, Shigeyuki Tada and Takeo Kanade

School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, Pennsylvania 15213-3890

Abstract— We have built a range-image sensor that acquires a complete 28×32 range frame in as little as one millisecond. Using VLSI, sensing and processing are combined into a unique sensing element that measures range in a fully-parallel fashion. The accuracy and repeatability of the sensed data is 0.1% or better. In this paper, we review the cell-parallel method used, describe our VLSI implementation, outline procedures for calibrating the cell-parallel sensor and present some experimental results. We conclude by describing a second-generation range sensor integrated circuit which is now being tested.

I. INTRODUCTION

A cell-parallel implementation greatly improves the performance of a light-stripe range-imaging sensor[1, 2, 3]. Though equivalent to conventional light-stripping from optical and geometrical standpoints, cell-parallel light-stripe sensors incorporate a fundamental improvement in the range measurement process. As a result, the acquired range data is more robust and more accurate. Furthermore, range image acquisition time is made independent of the number of data points in each frame. By fully exploiting the capability of VLSI to both sense and process information, we have built a smart sensor that acquires a complete frame of 10-bit range image data in a millisecond.

II. A CELL-PARALLEL APPROACH TO LIGHT-STRIPE RANGE IMAGING

Range information is crucial to many robotic applications. A range image is a 2-D array of pixels, each of which represents the distance to a point in the imaged scene. Many techniques for the direct measurement of range images have

This research was supported in part by an AT&T Foundation Grant, the National Science Foundation, under grant MIP-8915969, and the Defense Advanced Research Projects Agency, ARPA Order No. 7511, monitored by the NSF under grant MIP-9047590.

Published in the Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '92), Pages 349–58, Raleigh, NC, July 7–10, 1992.

0-7803-0737-2/92\$03.00/1992©IEEE

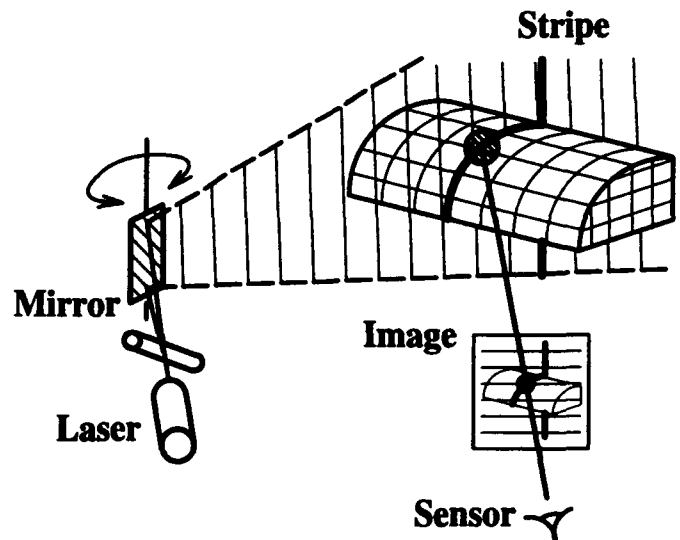


Fig. 1. Traditional light-stripe range imaging.

been developed[4]. Of these, the light-stripe methods have proven to be among the most robust and practical.

Fig. 1 illustrates the principle on which a light-stripe sensor is based. The scene to be imaged is lit by a stripe — a plane of light formed by fanning a collimated source in one dimension. The stripe is projected in a known direction using a precisely controlled mirror. When viewed by an imaging sensor, it appears as a contour which follows the profile of objects. The shape of this contour encodes range information. In particular, if projector and imaging sensor geometry are known, the distance to every point lit by the stripe can be determined via triangulation.

A conventional light-stripe range sensor builds a range image using a "step-and-repeat" procedure. A stripe is projected onto a scene, as described above, and one column of range image data is measured. The stripe is stepped to a new position and the process is repeated until the entire scene has been scanned.

Unfortunately, step-and-repeat implementations are slow. In order to build a complete range image using data from N stripe positions, N intensity images are required. The total time T_f^{Step}

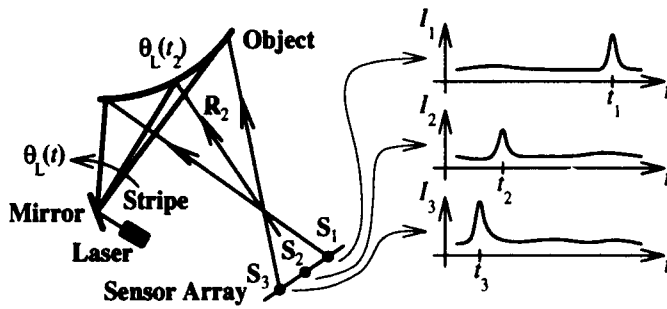


Fig. 2. Cell-parallel light-stripe range imaging.

to acquire the range frame is

$$T_f^{\text{Step}} = N T_f^{\text{Video}}. \quad (1)$$

Assuming $T_f^{\text{Video}} = 1/30$ second and $N = 100$, $T_f^{\text{Step}} = 3.3$ seconds is required.

The frame time of a step-and-repeat sensor has been improved by imposing additional structure on the light source. For example, the gray-coded sources used by Inokuchi[5] reduce the factor of N in (1) to $\log_2 N$. However, achievable frame rates are still too slow and the fundamental problem remains — range frame time increases with spatial resolution.

A. The Cell-Parallel Method

The cell-parallel technique is an elegant modification of the basic light-stripe algorithm. The technique is a dynamic one, with time an important aspect of the range measurement process[6].

Consider the geometry of a three-pixel, single-row cell-parallel range sensor, seen from above in Fig. 2. In the figure, the stripe plane is perpendicular to the page. The stripe is quickly swept across the scene from right to left, briefly illuminating object features. A sensing element, say S_2 , monitors the light intensity I_2 returned to it along a fixed line-of-sight ray R_2 . When the position of the stripe is such that it intersects R_2 at a point on the surface of an object, a "flash" will be observed by the sensing element.

Range to the object is measured by recording the time t_2 at which the flash is seen. The location of the stripe as a function of time is known because its projection angle $\theta_L(t)$ is controlled by the system. The "time-stamp" t_2 acquired by the sensing element measures the position of the stripe when its light is reflected back to the sensor. The three-dimensional coordinates of one object point are uniquely determined at the intersection of the line-of-sight ray R_2 with the stripe plane at $\theta_L(t_2)$ on the surface of the object.

A sensor which collects a dense range image is formed by arranging identical sensing elements into a two-dimensional array. The cells of the array work in parallel, gathering a range image during a single pass of the light stripe. The time

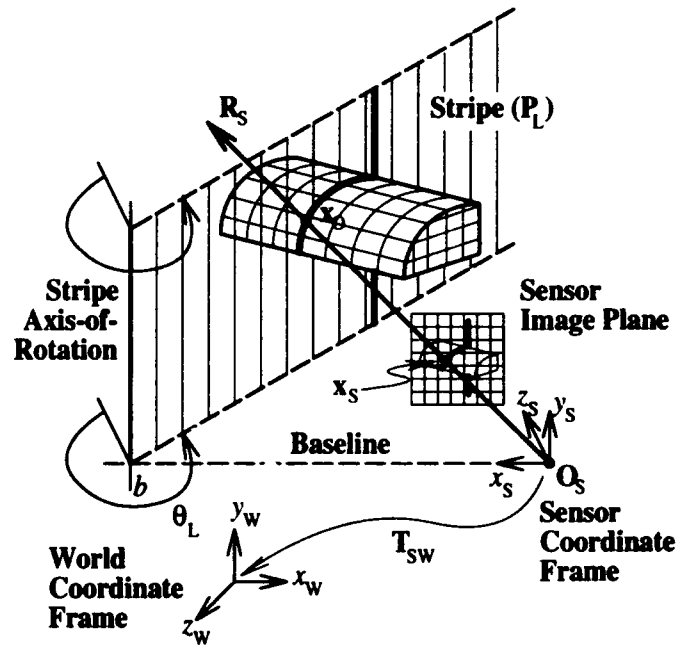


Fig. 3. Cell-parallel system geometry.

required to acquire the range frame is independent of its spatial resolution —

$$T_f^{\text{Cell}} = T_f^{\text{Stripe}}. \quad (2)$$

The frame time T_f^{Stripe} of a cell-parallel sensor is set by the bandwidth of the photo-receptor used in its sensing elements. Very high frame rates ($1/T_f^{\text{Stripe}}$) can be achieved. The photodiodes used in our cell design have bandwidth into the megahertz. They can detect a stripe moving at angular velocities in excess of 6,000 rpm.

B. Cell-Parallel System Geometry

Cell-parallel system geometry can be described using homogeneous coordinate transformations[7, 8]. Referring to Fig. 3, the origin of the frame O_s is placed at the optical center of the imager. The stripe is a half-plane which radiates out from an axis-of-rotation aligned with the y -axis of the frame and passing through the point

$$\mathbf{x}_L = [b \ 0 \ 0 \ 1]^T. \quad (3)$$

Stripe rotation θ_L is measured counter-clockwise about its axis when viewed from the positive y direction and defined to be zero when the stripe lies in the yz -plane. In a homogeneous representation, a plane is described in terms of a column vector \mathbf{P} that satisfies the scalar product $\mathbf{x}\mathbf{P} = 0$, where \mathbf{x} is a homogeneous point that lies in \mathbf{P} . In the sensor coordinate frame defined above, the stripe plane is modeled in terms of b and θ_L .

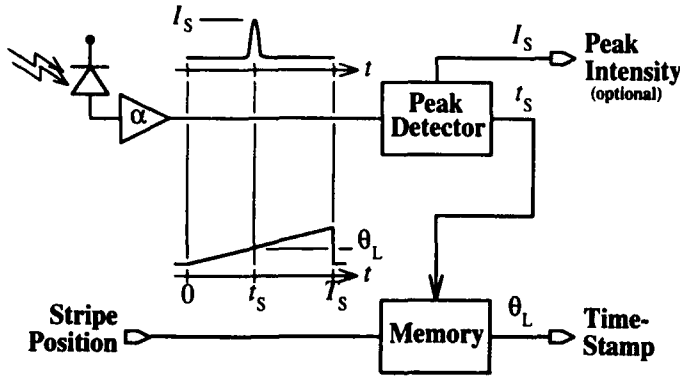


Fig. 4. Basic sensing element block diagram.

as

$$\mathbf{P}_L = \begin{bmatrix} -\cos \theta_L \\ 0 \\ \sin \theta_L \\ b \cos \theta_L \end{bmatrix}. \quad (4)$$

The position $\mathbf{x}_S = (x_S, y_S, z_S)$ of a sensing element on the sensor image plane defines the line-of-sight ray \mathbf{R}_S . The parametric equation for a line in three dimensions is used to represent \mathbf{R}_S as

$$\mathbf{x} = \frac{\tau}{\tau_S} (\mathbf{x}_S - \mathbf{O}_S) + \mathbf{O}_S \quad (5)$$

where $\tau_S = \|\mathbf{x}_S\| = \sqrt{x_S^2 + y_S^2 + z_S^2}$. The line parameter τ , when normalized by τ_S , is simply the distance along \mathbf{R}_S measured from \mathbf{O}_S heading toward the object.

The point of intersection \mathbf{x}_O , between the stripe and the line-of-sight, is found by solving $\mathbf{xP}_L = 0$ for τ :

$$\tau = \frac{b\tau_S}{x_S - z_S \tan \theta_L}. \quad (6)$$

In the coordinate frame of the sensor, this point is

$$\mathbf{x}_O = \left[\frac{\tau}{\tau_S} x_S \quad \frac{\tau}{\tau_S} y_S \quad \frac{\tau}{\tau_S} z_S \quad 1 \right]. \quad (7)$$

Thus, the 3-D position \mathbf{x}_O of imaged object points can be recovered from the scalar distance measurement τ .

III. VLSI RANGE SENSOR

A practical implementation of the cell-parallel range imaging algorithm requires a smart sensor — one in which optical sensing is local to the required processing. Silicon VLSI technology provided the means for building such a sensor.

Fig. 4 summarizes the operation of elements in the smart cell-parallel sensor array. Functionally, each must convert light energy into an analog voltage, determine the time at which the voltage peaks and remember the time at which the peak occurred.

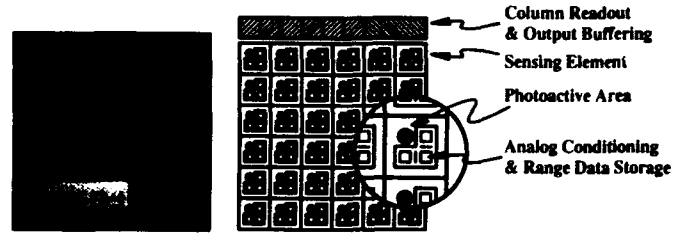


Fig. 5. Range sensor integrated circuit.

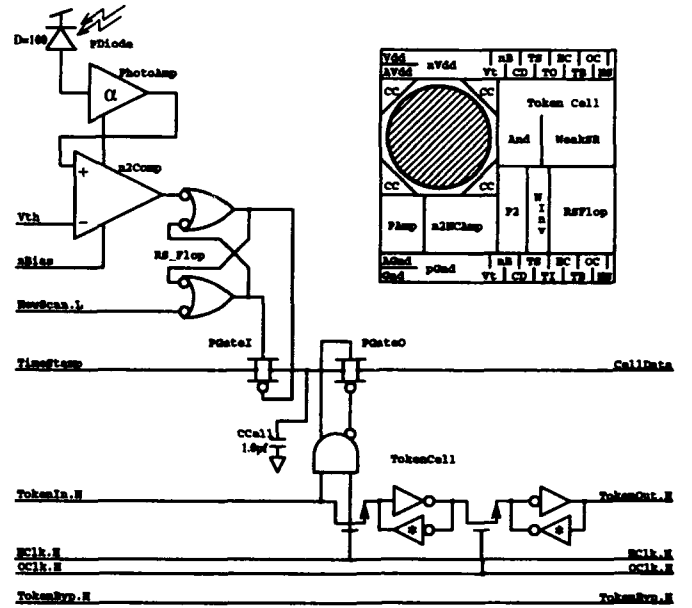


Fig. 6. Sensing element circuitry.

A. A 28 x 32 Cell-Parallel Sensor Chip

The multi-pixel cell-parallel range sensor we have developed is shown in Fig. 5. This chip consists of 896 sensing elements arranged in a 28 x 32 array. It was fabricated using a 2 μm p-well CMOS, double-metal, double-poly process and measures 9.2 mm x 7.9 mm (width x height). Of the total 73 mm² chip area, the sensing element array takes up 59 mm², read-out column-select circuitry 0.37 mm² and the output integrator 0.06 mm². The remaining 14 mm² is used for power bussing, signal wiring, and die pad sites.

B. Sensing Element Design

The architecture chosen for the range sensing elements is shown in Fig. 6. Areas of interest in the diagram include the photo-receptor (PDiode), the photo-current transimpedance amplifier (PhotoAmp), threshold comparison stage (n2Comp), stripe event memory (RS_Flop), time-stamp track-and-hold circuitry (PGateI/CCell) and cell read-out logic (PGateO/TokenCell).

In operation, sensing elements cycle between two phases —

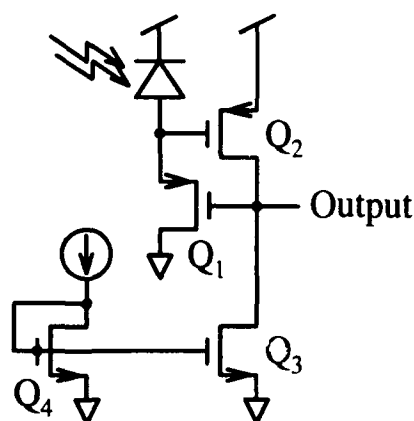


Fig. 7. Non-linear transimpedance amplifier.

acquisition and read out.

During the acquisition phase, each sensing element implements the cell-parallel procedure of Fig. 4. The photodiode within a cell monitors light energy reflected back from the scene. Photocurrent output is amplified and continuously compared to an external threshold voltage V_{th} . When photoreceptor output exceeds this threshold, the "stripe-detected" latch in the cell is tripped. The value of the time-stamp voltage at that instant is held on the capacitor C_{CELL} , recording the time of the stripe detection.

The acquisition phase is synchronized with stripe motion and ends when the stripe completes its scan. At that time, the array sensing elements recorded a range image in the form of held time-stamp values. This raw range data must now be read from the chip.

A time-multiplexed read-out scheme off loads range image data in raster order through a single chip pin. One bit of token state is passed through the sensing element array, selecting cells for output. Dual n/p -transistor pass gate structures are used throughout the time-stamp data path. They permit the use of rail-to-rail time-stamp voltages, maximizing the dynamic range of the analog time-stamp data.

C. Stripe Detection

One of the more challenging aspects of the cell design involved the circuitry which detected the stripe.

A photodiode forms the light sensitive area within each cell. This diode is a vertical structure, built using the n -substrate as the cathode and the p -well of the CMOS process as the anode. An additional p^+ implant, driven into the well, reduces the surface resistivity of the anode and increases the device bandwidth.

The non-linear transimpedance amplifier of Fig. 7 was a key element of the sensor cell design. Reflected light from the swept stripe source generates nano-amp photo-current pulses and thus a very high-gain amplifier is required to convert this current into a usable voltage. In addition, very little die area

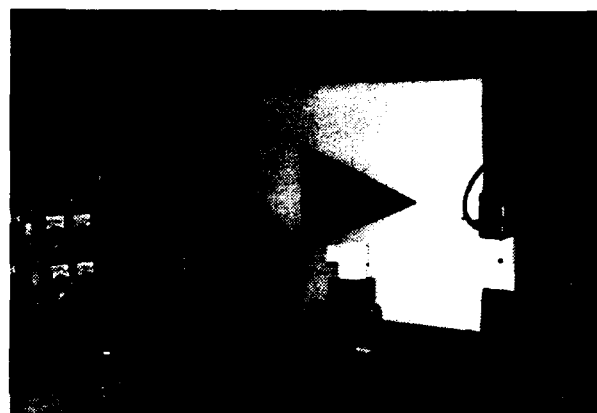


Fig. 8. The cell-parallel range-finding system.

could be devoted to photo-current amplification if cell area was to be kept small. The three transistor amplifier design of Fig. 7 satisfies both requirements. Its logarithmic transfer characteristic provides freedom from output saturation even when input light levels vary over several orders of magnitude. The output rise-time of photodiode/amplifier test structures in response to a stripe was measured to be a few microseconds.

D. Analog Signal Processing

Analog signal processing techniques played an important role in the design of this smart sensor. As shown in Fig. 6, sensing elements use analog circuitry to amplify the photo-current, to detect the stripe and to record the per-cell time-stamp information. Stripe timing is represented in analog form as a 0-5 V sawtooth broadcast to all cells of the array. This allowed the time-stamp value to be stored as charge on the 1 pf capacitor within each cell. The digital equivalent of latching a count into a multi-bit register would be significantly larger in area and would require that the digital time-stamp counters run during the acquisition phase. Thus, analog processing kept cell area small and minimized digital switching noise during photo-current measurements in the acquisition phase.

IV. PROTOTYPE RANGE IMAGE SENSOR

The 28×32 element VLSI sensor prototype described in the previous section was incorporated into the light-stripe range system shown in Fig. 8. System components visible in the photograph include (from the left) the stripe generation assembly, the VLSI sensor chip and its interface electronics, a calibration target and the 3-DOF positioning system. Table I provides details of the configuration shown.

V. CELL-PARALLEL SENSOR CALIBRATION

Calibration provides the complete specification of system geometry necessary for converting cell time-stamp data into range

TABLE I
CEL: PARALLEL SENSOR SYSTEM SUMMARY

Baseline	300 mm
Laser Source	Laser Diode (Collimated)
Wavelength	780 nm
Output Power	30 mW
Stripe Width	1 mm
Stripe Spread	40° (3 dB)
Sweep Assembly	Rotating Mirror
Sweep Angle	40°
Sensor Optics	1/2"-Format CCD Zoom Lens
Focal Length	12.5 to 75 mm
f-number	f/1.8
A/D Precision	12 bits

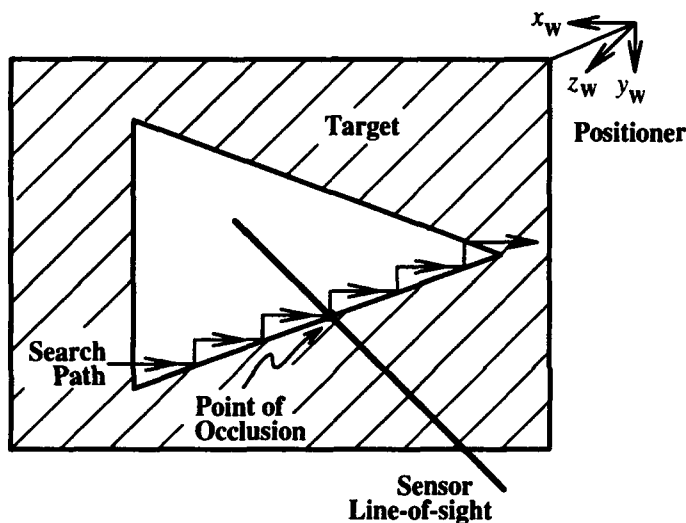


Fig. 9. Line-of-sight measurement.

images. Two sets of calibration parameters must be measured. First, 3-D sensor chip geometry and optical parameters must be measured — the *imager model*. Next, a mapping between time-stamp values θ_s and distance r for all sensing elements is developed — the *stripe model*.

A. Imager Model Calibration

This method measures component model geometry using reference objects, manipulated in the sensor's field of view with an accurate 3-DOF (degree of freedom) positioning device. The following two-step procedure is used (Fig. 3):

- the line-of-sight rays R_s for a few cells are measured, and
- a pinhole-camera model is fit to measured line-of-sight rays in order to approximate line-of-sights for all sensing elements.

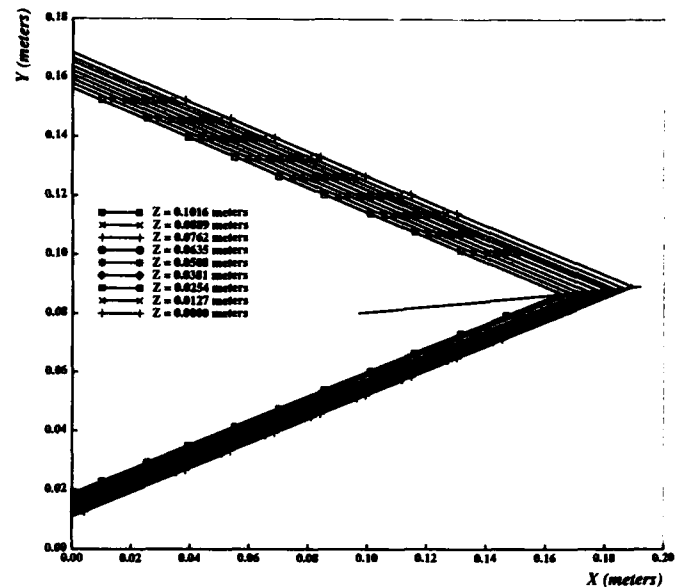


Fig. 10. Cell (13,15) measured line of sight.

A planer target out of which a triangular hole has been cut as shown in Fig. 9 is used to map out sensing element line-of-sight rays. The target is mounted on the positioner so that its surface is parallel to the world- xy plane.

A single 3-D point on the line-of-sight of a particular sensing element is found as follows. The target is moved to some z -position in world coordinates and held. The bottom edge of the triangular hole is located by moving the target around in x and y as indicated in Fig. 9. When a small motion in either x or y causes a large change in the time-stamp value reported by the cell, occlusion of the line-of-sight at an edge of the triangular cut is indicated.

Once many points along the bottom edge are located, a line, known to lie in the plane of the target, is fit. The location of the top edge is found in a similar fashion. The intersection of the top and bottom edge lines define one 3-D point that lies on the cell's line-of-sight. A number of these points are located by moving the target in z and repeating the process. The line-of-sight for a single cell can then be identified by fitting a 3-D line to these points. Experimental data from the calibration of one sensing element's line-of-sight is shown in Fig. 10.

Mapping the line-of-sight rays for all 896 sensing elements in this manner is too time consuming. In practice, line-of-sight information is measured for 25 cells, evenly spaced in a 5 grid. The geometry of the remaining cells is approximated using a pinhole-camera model.

The pinhole-camera model[11] constrains all sensing element line-of-sight rays to pass through a single point focus of expansion at the optical center of the camera. Fig. 11 graphically illustrates the process. Sensing element locations are assumed to lie in some *sensor plane*, at locations evenly spaced in a 2-D grid on the plane. Eleven model parameters must be

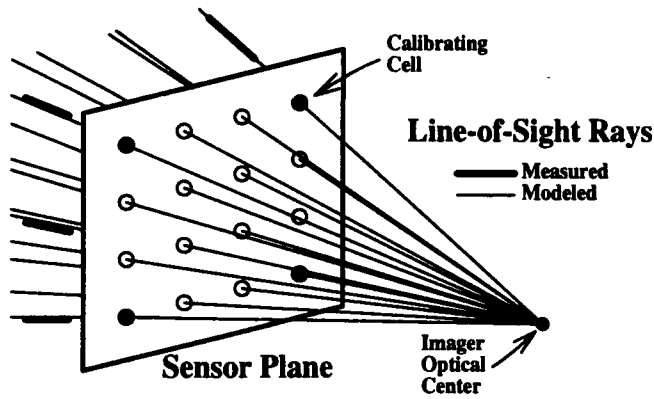


Fig. 11. "Pinhole" line-of-sight approximation.

determined that identify the transformation matrix T_{SW} and the geometry of the the sensor plane. A least-squares procedure is used to fit pinhole-model parameters to line-of-sight information measured in the first calibration step. Imager model geometry is now fully calibrated.

B. Advanced Imager Model Calibration

Unfortunately, calibration of the imager model via line-of-sight measurement is not suitable for use outside of the laboratory environment. "One-at-a-time" measurement of sensing element geometry, as outlined above, is slow and cumbersome.

We are developing a faster, more precise method for imager model calibration. In this new calibration method, the 3-DOF positioning system is replaced with a liquid crystal display (LCD) mask that need only be accurately positioned along one degree of freedom. The LCD mask is used to define precise black-and-white images that are "seen" by the range sensor. The method relies on intensity image information, measuring geometry through analysis of reference object images[9].

The LCD mask is placed between a diffuse planar target and sensor chip at a known position and is backlit by shining the system stripe source on the planar target. The pattern displayed on the LCD forms a black-and-white image on the sensor. Only illuminated sensing elements will latch the stripe-detected condition (Section III-B). A single-bit intensity image is derived by identifying the time-stamp output of illuminated sensing elements.

Sensing element line-of-sight geometry is found by varying the LCD mask pattern in a controlled fashion. For example, a circular pattern, whose 3-D center is known, can be projected. A calibration point is found by measuring the 2-D location of this circle's center in the intensity image returned by sensor. Additional calibration data is measured by varying the position of the circle on the LCD mask and the position of the LCD along z_s . Also, by measuring the center different radii of the circle at a fixed position, we can compensate for the low spatial resolution of the current sensor. The new sensor chip design,

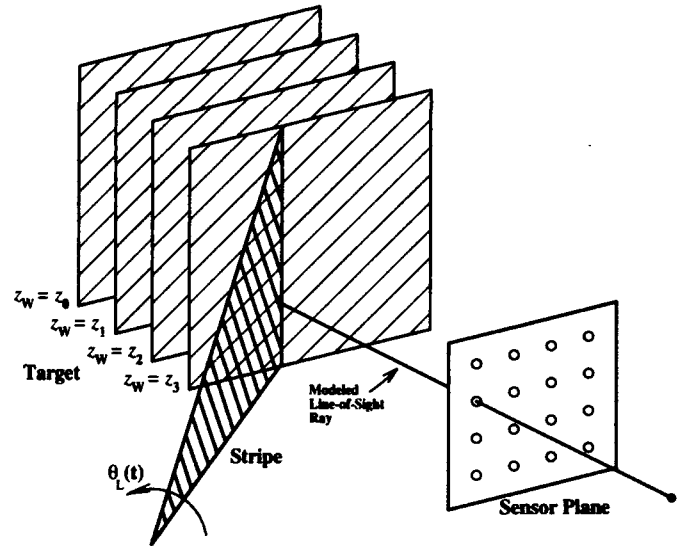


Fig. 12. Time-stamp calibration.

discussed in Section VII, returns multi-bit intensity image data which further assists imager geometry calibration.

Use of the LCD mask significantly reduces the time required to perform imager-model calibration. In the previous method, two edges of a triangular hole had to be mapped out, via accurate back-and-forth movement, in order to yield a single calibration point. In the new method, one calibration point is measured from a single LCD-generated pattern without mechanical X - Y movement. Precise calibration of the low-spatial resolution range sensor is possible because high-precision patterns are generated by the LCD mask.

The use of an LCD mask to project precise 2-D patterns has application beyond the calibration of our light-stripe range sensor. For example, this technique could be used to assist more traditional camera calibration procedures or to present training data to image-based neural net systems. LCD displays have several advantages over CRT displays for applications like these — they are fast, they are static (not refreshed), and they form images which are stable and well defined.

C. Stripe Model Calibration

The second part of the calibration procedure determines the mapping between time-stamp data and range along all sensing element line-of-sight rays. As shown in Fig. 12, a planar target with no hole replaces the target used in step one. The new target is held at a known world- z position, parallel to the xy plane, and time-stamp readings θ_s from all sensors are recorded. This process is repeated for many z positions. Using this information, the function which maps cell time-stamp values θ_s into line-of-sight distance τ for each sensing element is approximated by fitting a parabola to each. Experimental data, showing the fitted τ versus θ_s functions for several sensing elements, is shown in Fig. 13. Calibration of the cell-parallel

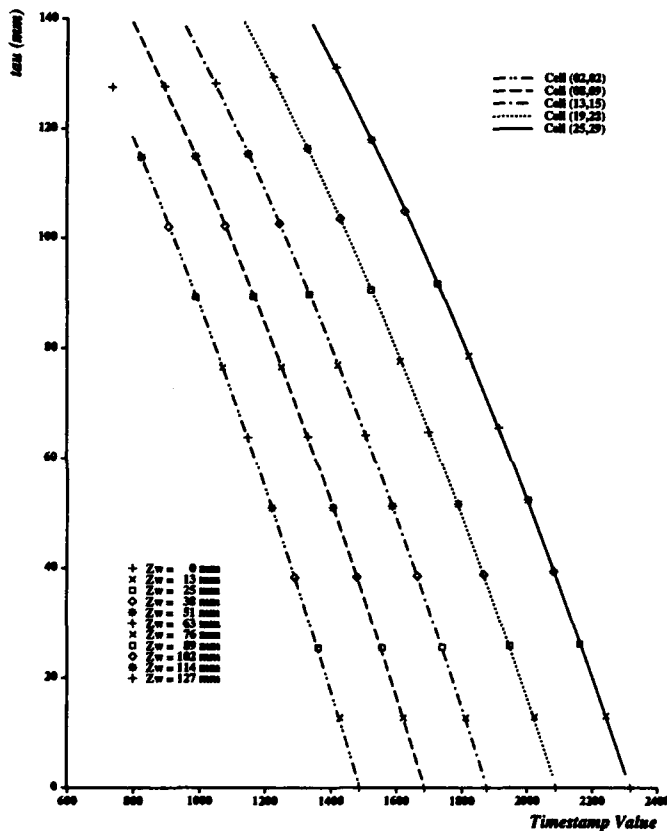


Fig. 13. Time-stamp calibration result.

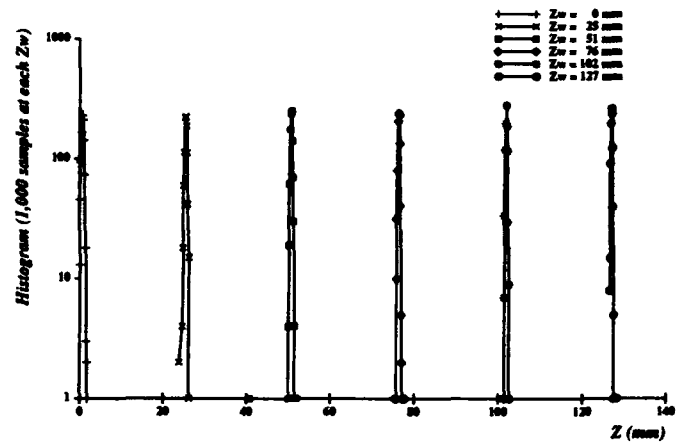


Fig. 14. Cell (13,15) range-data histograms.

range sensor is now complete.

VI. SYSTEM PERFORMANCE

A. Range Accuracy and Repeatability

The quality of the range data produced by the cell-parallel range sensor was measured by holding a planar target at a known world- z position with the 3-DOF positioning device. In the experimental setup, the world- z axis heads almost directly toward the sensor with the $z_w = 0$ point roughly 500 mm away. Analog time-stamp values from the sensor array were digitized, using a 12-bit analog-to-digital converter (A/D), and recorded for 1,000 trials. Light-stripe sweep (acquisition phase) time for each scan was 3 msec.

A histogram of the range data reported by one cell is plotted in Fig. 14. The horizontal axis represents the digitized time-stamp value, converted to world- z distance via the calibration model. Data for six world- z positions are combined in this plot. The vertical axis shows the number of times (plotted logarithmically), out of the 1,000 trials, that the sensing element reported that world- z distance. The sharpness of each peak is an indication of the stability (repeatability) of the range measurements.

Averaged statistical data for 25 evenly-spaced sensing elements is plotted in Fig. 15. In order to measure accuracy and repeatability, the position of the target, as reported by the cell-parallel sensor, is compared to the actual target z position. The "boxed" points in the plot represent the mean absolute error, expressed as a fraction of the world- z position and averaged for the 25 elements at z_w . One standard deviation of "spread", also normalized with z_w , is shown ($\bar{\sigma}$) above and below each box.

The experiments show the mean measured range value to be within 0.5 mm at the maximum 500 mm z — an accuracy of 0.1%. The aggregate distance discrepancy between world and measured range values remains less than 0.5 mm over the entire

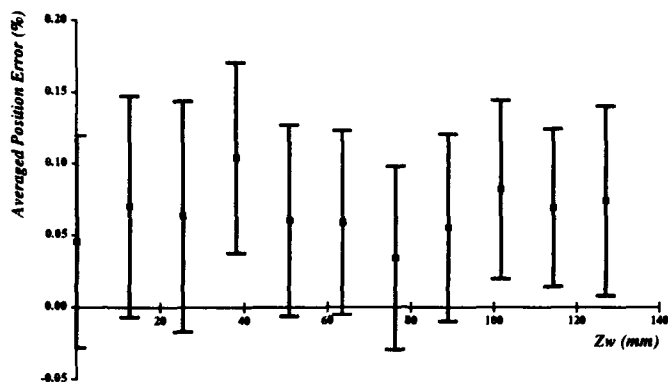


Fig. 15. Range data accuracy and repeatability.

360 mm to 500 mm z range. The cell-parallel sensor repeatability is found by computing the standard deviation of the distance measurements. The measured repeatability of histogram data is less than 0.5 mm — 0.1% at the maximum 500 mm positioner translation. The 0.5 mm repeatability decreases with the distance to the sensor — essentially with the slope of the time-stamp to distance mapping function (Fig. 13).

B. Range Image Acquisition

Fig. 16 shows a wire-frame representation of one 28×32 range image produced by the sensor. The imaged object is the cup shown in the figure, approximately 80 mm in diameter at its opening and 80 mm high. The range sensor is looking directly at the object from a distance of 500 mm. The viewpoint of the plot is at a point directly above the optical center of the sensor. The complete range image was acquired during a 3 msec stripe scan. The intersection points of the wire-frame plot are positioned on cell line-of-sight rays at the measured distance along the ray and the focus of expansion is located in front of the cup. Thus, the smaller "squares" represent object surface patches closer to the sensor. This is opposite the manner in which straight perspective would make an object with a grid painted on it appear, and at first glance gives the false impression that the "mold" used to make the cup has been imaged.

The curved smooth front surface of the object is clearly visible in the range data. The 20 mm handle of the cup is readily distinguished, as is the planar background behind the cup. The curved surface of the object halfway down the cup directly across from the bottom of its handle includes a slight shift of the wire-frame. The imaged cup is slightly narrower at its base by about 2 mm. The cell-parallel sensor is measuring this small 3-D feature at the 500 mm object distance.

C. Sensor Performance Summary

A summary of the cell-parallel sensor system performance is given in Table II.

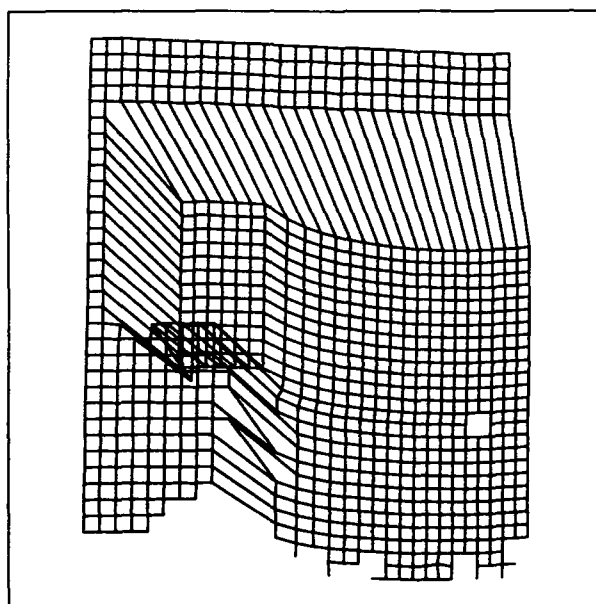
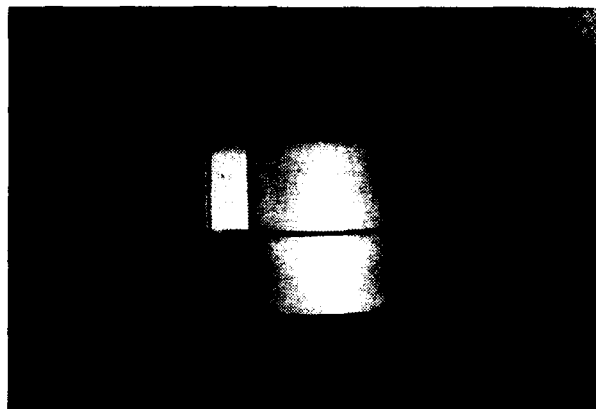


Fig. 16. Range data wire frame.

TABLE II
CELL-PARALLEL SENSOR PERFORMANCE SUMMARY

Spatial Resolution	28×32
Frame Time	Up to 1 msec
Operating Distance	350 to 500 mm
Accuracy	< 0.5 mm
Repeatability	< 0.5 mm

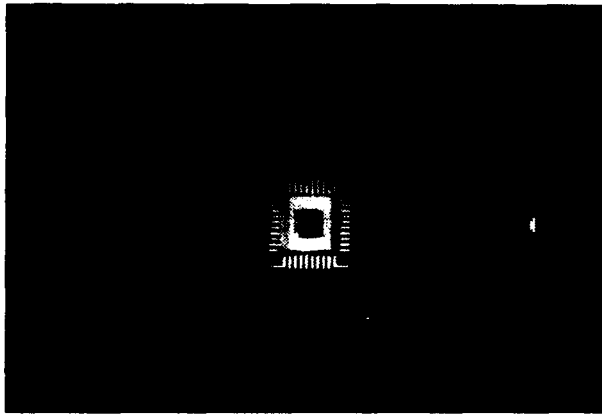


Fig. 17. Second-generation range sensor integrated circuit.

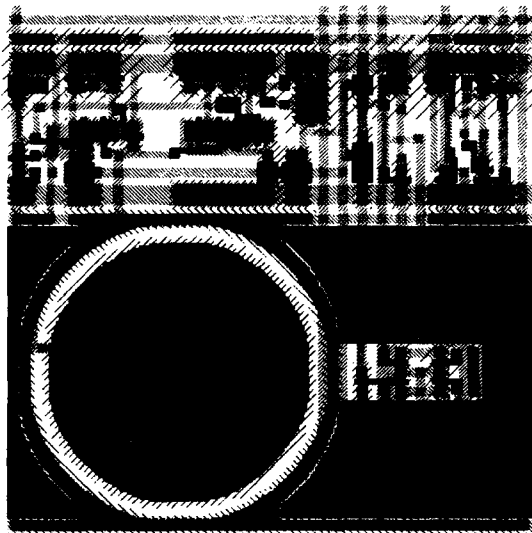


Fig. 18. Second-generation sensing element layout.

VII. A SECOND GENERATION SENSING ELEMENT

A second-generation implementation of the light-stripe sensor array has been fabricated. This new chip, seen in Fig. 17, incorporates several advantages over the first design. The die area of the new cell, shown in Fig. 18, is $216\mu\text{m} \times 216\mu\text{m}$, 40% smaller than that of the cells of the first-generation sensor (photoreceptor area has been kept constant). Stripe detection is done in a more robust manner and range data read-out circuitry has been simplified. In addition, the new cell provides a means to record and read out the value of the peak intensity seen when it acquires a range data sample. The peak intensity information provides a direct measure of scene reflectance because stripe output power is known and distance to the object point is measured. In addition, the availability of intensity information allows for efficient sensor calibration (Section V-B).

Peak detection is done using the circuit of Fig. 19. Operation of the circuit is straightforward. The source following transi-

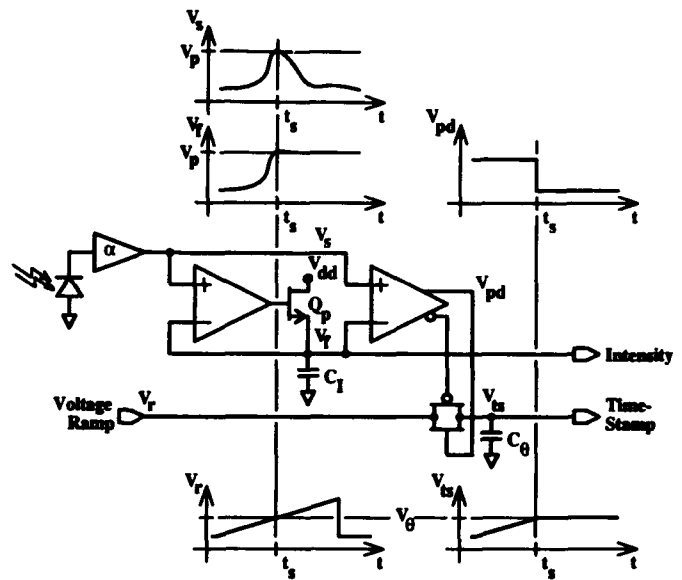


Fig. 19. Second-generation sensing element circuitry.

tor Q_p enables capacitor C_1 to track the rising intensity input voltage transitions. No path is provided for C_p to discharge when photoreceptor output transitions downward. At the end of a scan, the largest intensity reading observed will be held. Stripe detection is easily accomplished by comparing the peak-intensity value V_f with the amplified photodiode output V_s . When V_s falls below the V_f , the output from the comparator is used to record a time-stamp value.

Using *Spice*[10], operation of the second-generation sensing element design was simulated. The simulation results are plotted in Fig. 20. The output from the peak-following circuit *XLSCCELL.30* acts as a dynamic threshold for each cell, replacing the externally applied global threshold of the first-generation design (Section III-B). Comparator input offset mismatch made setting a global threshold level, valid for all cells in the array, difficult. Thus, stripe detection is made more robust by this modification. In addition, the "true" peak detection of the new design provides better quality range data because the new stripe detection scheme identifies the location of the peak in time more accurately than simple thresholding.

The peak-intensity value held within the second-generation cell is an important artifact of the ranging process and, in the new design, is provided as an additional sensing element output. The illumination source in the system, the stripe, is of known power. Intensity reduction from $1/r$ -type losses can be accounted for because range to the object is measured. The intensity value therefore provides a direct measure of scene reflectance properties at the stripe wavelength. It is an image aligned perfectly with range readings from the cell array.

The area in each cell dedicated to time-stamp read out is much smaller in the new design. Direct addressing of the cell to be read, using row and column selects, eliminates the token

REFERENCES

- [1] A. Gruss, *A VLSI Smart Sensor for Fast Range Imaging*. PhD thesis, Carnegie Mellon University, November 1991.
- [2] T. Kanade, A. Gruss, and L. R. Carley, "A very fast VLSI rangefinder," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, (Sacramento, CA), pp. 1322-29, April 1991.
- [3] A. Gruss, T. Kanade, and L. R. Carley, "Integrated sensor and range-finding analog signal processor," *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 184-191, March 1991.
- [4] P. J. Besl, "Range imaging sensors," Research Publication GMR-6090, General Motors Research Laboratories, March 1988.
- [5] S. Inokuchi, K. Sato, and F. Matsuda, "Range imaging system for 3-D object recognition," in *Proceedings of 7th International Conference on Pattern Recognition*, (Montreal, Canada), pp. 806-808, July 1984.
- [6] K. Araki, Y. Sato, and S. Parthasarathy, "High speed rangefinder," in *Optics, Illumination, and Image Sensing for Machine Vision II*, vol. 850, pp. 184-188, SPIE, 1987.
- [7] D. H. Ballard and C. M. Brown, *Computer Vision*. Prentice-Hall, Inc., 1982.
- [8] W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics*. McGraw-Hill Book Company, 2nd. ed., 1979.
- [9] M. D. Altschuler, K. Bae, B. R. Altschuler, J. T. Di-jak, L. A. Tamburino, and B. Woolford, "Robot vision by encoded light beams," in *Three-Dimensional Machine Vision* (T. Kanade, ed.), Kluwer Academic Publishers, 1987.
- [10] Meta-Software, Inc., 1300 White Oaks Road, Campbell, CA 95008, *HSPICE User's Manual*, h9001 ed., 1990.
- [11] J. Weng, P. Cohen, and M. Herniou, "Calibration of stereo cameras using a non-linear distortion model," in *Proceedings of the 10th International Conference on Pattern Recognition*, (Atlantic City, NJ), pp. 246-253, IEEE Computer Society Press, June 1990.

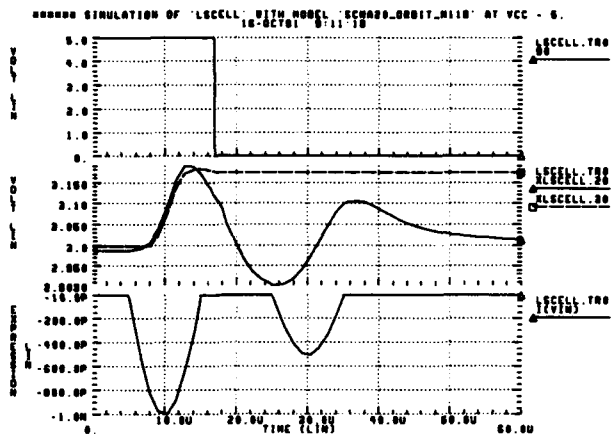


Fig. 20. Second-generation sensing element simulation result.

state necessary in the first-generation design. The $N \times M$ array is read using N row select lines and M column select lines. A given cell is enabled for read out by asserting the row and column select lines that correspond to the location of the cell in the array. The two-level bus hierarchy has been maintained, however, to keep bus loading at a minimum. The area savings of the new read selection method has made cell area of the second-generation design smaller despite the additional peak detection circuitry.

VIII. CONCLUSION

We have presented the design and construction of a very high-performance range-imaging sensor. This sensor acquires a complete 28×32 range-data frame in a few milliseconds. Its range accuracy and repeatability were measured to be less than 0.5 mm on average at half-meter distances. The success of this implementation can be attributed to the use of a VLSI smart sensor methodology that allowed a practical implementation of the cell-parallel technique.

While the advantages of processing at the point sensing have been advocated by many, few practical smart-sensor implementations have been demonstrated. The cell-parallel range imager presented here bridges the gap between smart sensor theory and practice, demonstrating the impact that the smart sensor methodology can have on robotic perception systems, like automated inspection and assembly tasks.

Smart VLSI-based sensors, like the high-speed range imager sensor presented here, will be key components in future industrial applications of sensor-based robotics.

3-D Stereo Using Photometric Ratios

Lawrence B. Wolff

Elli Angelopoulou

Computer Vision Laboratory
Department of Computer Science
The Johns Hopkins University
Baltimore, Maryland 21218

Abstract

We present a novel robust methodology for corresponding a dense set of points on an object surface for 3-D stereo computation of depth. The methodology utilizes multiple stereo pairs of images, each stereo pair taken of exactly the same scene but under different illumination. With just 2 stereo pairs of images taken respectively for 2 different illumination conditions, a stereo pair of ratio images can be produced; one for the ratio of left images, and one for the ratio of right images. We demonstrate how the photometric ratios composing these images can be used for accurate correspondence of object points. Object points having the same photometric ratio with respect to 2 different illumination conditions comprise a well-defined equivalence class of physical constraints defined by local surface orientation relative to illumination conditions. We show how for diffuse reflection the photometric ratio is invariant to varying camera characteristics and viewpoint and that therefore the same photometric ratio in both images of a stereo pair implies the same equivalence class of physical constraints. Corresponding photometric ratios along epipolar lines in a stereo pair of images under different illumination conditions is therefore a robust correspondence of equivalent physical constraints, and determination of depth from stereo can be performed without knowing what these physical constraints being corresponded actually are. This implies a very practical shape-from-stereo methodology applicable to perspective views and not requiring any knowledge whatsoever of illumination conditions. This is particularly practical for determination of 3-D shape on smooth featureless surfaces which has previously been hard to perform using stereo. We demonstrate experimental 3-D shape determination from a dense set of points using our stereo technique on smooth objects of known ground truth shape that can be accurate to well within $\pm 1\%$ relative depth.

1 INTRODUCTION

There has been extensive work on computational stereo vision including [11], [10], [4], [14], and [1]. A large collection of articles on stereo are contained in the recent book by Mayhew and Frisby [12]. Much of the work computing depth from stereo vision involves the correspondence of image features such as intensity discontinuities or zero crossings determining image edges. These are features that can be computed directly from an image without any knowledge of the image formation process. A possible disadvantage of depth determination from 3-D stereo using edges is that this data can be sparse and a number of methods have been developed to interpolate smooth surfaces to sparse depth data from stereo [4],

[17]. There are considerable problems with shape determination of smooth featureless objects using feature-point based stereo vision algorithms.

Grimson [5] was the first to consider utilizing the reflectance properties of surfaces to augment shape determination from stereo vision. Utilizing diffuse shading information from two camera views Grimson determined surface orientation at zero crossings using this in addition to depth information at these points to more accurately interpolate a surface. Diffuse reflection was assumed to be Lambertian and a Phong [13] specular component was also assumed to exist. Local surface orientation was determined by a modified photometric stereo technique [20] adapted to binocular stereo.

Smith [16] considered the correspondence of points in a stereo pair of images of a smooth featureless Lambertian reflecting surface utilizing a mathematical formulation he termed the *Stereo Integral Equation*. What is unique about this work is that except for knowing *a priori* the correspondence of endpoints along an epipolar line, all points can be properly corresponded between these endpoints purely from photometric values. This in turn provides for a very dense depth map from stereo vision.

There has been work by Blake, Brelstaff, Zisserman and others [2], [3], [21] that has exploited the geometry of specular reflection viewed from a stereo pair of cameras to derive constraints on surface shape. This work however depends upon the correspondence of segmented specularities rather than the correspondence of actual photometric values, which is the primary concern of this paper.

The major advantage of being able to accurately correspond photometric values between a stereo pair of images, besides being able to determine the shape of smooth featureless surfaces, is that this would provide for a very dense depth map. There are a number of practical issues concerning stereo vision which makes this very difficult. First and foremost is that stereo vision requires 2 cameras, and no two cameras even if they are exactly the same model number, records image intensities exactly the same. Even if a surface were perfectly Lambertian reflecting so that reflected radiance is completely independent of viewpoint, because of camera problems and the details of image formation described in the Problem Background section, the same reflected radiance from an object point will be recorded with an unpredictably different gray value for each camera. In fact, diffuse reflection is not even Lambertian as such reflection is actually dependent upon viewing angle. Specular reflection is observed at different object points from different views which further complicates the situation.

We present a novel practical methodology for reliably

corresponding photometric values in a stereo pair of images that overcomes most of the problems inherent to using a stereo pair of cameras without having to perform a large amount of camera calibration. We need the additional requirement of at least 2 illumination conditions but never need to know anything about these illumination conditions. Our methodology corresponds the left and right ratio images of the same scene under arbitrarily different illumination conditions. We prove that the photometric ratios arising from diffuse reflection are invariant to most characteristics varying between a stereo pair of cameras as well as invariant to viewpoint and diffuse surface albedo, that make pixel gray values by themselves unreliable for stereo correspondence. We also show that corresponding photometric ratios is equivalent to corresponding classes of well-defined physical constraints on object points. Furthermore, correspondence of photometric ratios can be done to subpixel accuracy using interpolation.

A technique that is somewhat related to our stereo methodology is "dual photometric stereo" pioneered by Ikeuchi [8]. There are however major conceptual and implementation differences. The idea of dual photometric stereo is to apply photometric stereo [20] to a smooth object surface from two camera views. Surface orientation estimates produced from photometric stereo are segmented according to where they fall on a tessellated Gaussian sphere. Segmented orientation classes are corresponded between the stereo pair of images using area, mean surface orientation, and, epipolar constraints. A coarse depth map is computed, and then refined using an iterative scheme that forces the gradient of the depth map to be consistent with surface orientation. While Ikeuchi corresponds surface orientation estimates produced from multiple illumination in a stereo pair of images, our methodology corresponds equivalence classes of physical constraints between a stereo pair of images without ever having to compute these physical constraints explicitly. Ikeuchi does have the extra information provided by surface orientation to refine his depth map, but this is restricted to nearly orthographic views and known incident orientation of at least 3 distant light sources. The 3-D stereo method using multiple illumination that we are proposing never requires knowledge of any of the multiple illumination conditions and is applicable to perspective views.

2 Problem Background

We describe the problematic issues of comparing image intensities between a stereo pair of cameras. To do this we need to understand about the image formation process and the nature of reflection from objects.

We describe the formation of image intensity values beginning with the familiar relation from Horn and Sjöberg [7]:

$$E = L_r(\pi/4)(D/i)^2 \cos^4 \alpha, \quad (1)$$

relating image irradiance, E , to reflected radiance, L_r . The lens diameter, D , image distance, i , and light angle, α , incident on the camera lens are depicted in Figure 1. Equation 1 assumes ideal pinhole optics. The effective diameter, D , of a lens can be controlled with an aperture iris the size of which is measured on an F-stop scale. As F-stop is the ratio of focal length to effective lens diameter, image irradiance is inversely proportional to the square of the F-stop value on a camera lens. Image irradiance is therefore very sensitive to F-stop. While a stereo pair of cameras can use identical lens models at exactly the same

F-stop setting, the effective lens diameters can still be slightly different. The focal lengths as well can be slightly different and recalling the classical thin lens law

$$\frac{1}{i} + \frac{1}{o} = \frac{1}{f},$$

this will influence the image distance, i , in turn effecting the image irradiance. Even in the ideal case where focal lengths are precisely equal, the image distance, i , can be slightly different for a stereo pair of images even though the images "appear" equivalently in focus when in fact they are not precisely in focus. On top of all this is the dependence of image irradiance in perspective images on pixel location relative to the optical center of the image plane. The farther a pixel is radially away from the optical center, the larger is the light incident angle, α , which strongly effects image irradiance. Image irradiances arising from the same object point appear in different parts of a stereo pair of images making them difficult to compare.

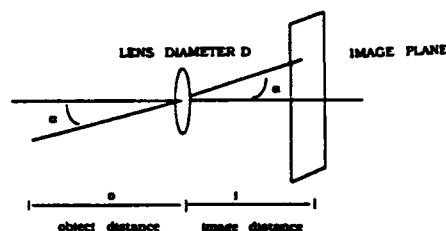


FIGURE 1

Equation 1 only takes into account the optics involved in image formation. Image irradiance is converted into pixel gray value using electronics. In general the conversion of image irradiance, E , into pixel gray value, I , can be described by the expression

$$I = gE^{\frac{1}{\gamma}} + d, \quad (2)$$

where g is termed the gain, d is the dark reference, and, γ , controls the non-linearity of gray level contrast. It is typically easy to set $\gamma = 1.0$ producing a linear response, and easy to take a dark reference image with the lens cap on, then subtracting d out from captured images. However, we have observed that not only can the gain, g , be variable between identical model cameras but this can change over time especially for relatively small changes in temperature. Unless g is calibrated frequently, comparing pixel gray values for identical image irradiances between a stereo pair of cameras can be difficult.

A widely used assumption about diffuse reflection from materials is that they are Lambertian [9] meaning that light radiance, L , incident through solid angle, $d\omega$, at angle of incidence, ψ , produces reflected radiance:

$$L \rho \cos \psi d\omega$$

independent of viewing angle. The independence of diffuse reflected radiance with respect to viewing angle makes it theoretically feasible to associate radiance values with object points in a stereo pair of images. However, even for ideal Lambertian diffuse reflectance the above discussion outlines the practical difficulties in achieving an accurate correspondence of pixel gray values produced from reflected radiance in a stereo pair of cameras.

The physical reality of diffuse reflection makes it even more practically difficult to associate diffuse reflected radiance with object points across a stereo pair of images. A recently proposed diffuse reflectance model for smooth

dielectric surfaces [18], [[19] these proceedings] empirically verified to be more accurate than Lambert's Law, expresses the dependence of diffuse reflected radiance on both angle of incidence, ψ , and viewer angle, ϕ , (see Figure 2), as

$$L \rho [1 - F(\psi, n)] \cos \psi [1 - F(\sin^{-1}(\frac{\sin \phi}{n}), 1/n)] d\omega \quad (3)$$

where the functions, $F()$, are the Fresnel reflection coefficients [15], and, n , is the index of refraction of the dielectric surface, and, ρ , is the diffuse albedo. Figure 3 shows the significant dependence of diffuse reflection upon viewer angle. Diffuse reflected radiance from an object point as seen from the two different viewpoints of a stereo pair of cameras will almost always not be equal.

The dependence of specular reflection upon viewpoint is even more severe due to its highly directional nature and the geometry of angle of incidence equals angle of reflection.

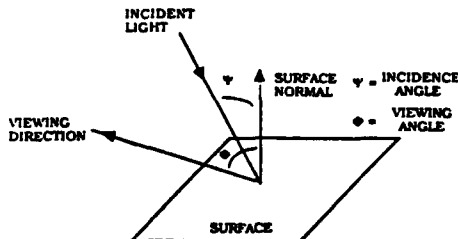


FIGURE 2

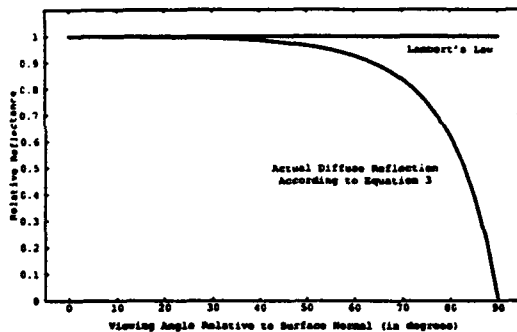


FIGURE 3

3 Using Photometric Ratios For 3-D Stereo

The discussion and analysis of the previous section has shown that pixel gray values by themselves are unreliable in being associated with object points in a well-defined way for correspondence between a stereo pair of images. We show that the ratio image produced from 2 images of diffuse reflection from the same scene respective to 2 different (but not necessarily known) illumination conditions is invariant to the differences in physical characteristics of cameras discussed in the previous section, as well as viewpoint and diffuse surface albedo. Furthermore, these photometric ratios can be associated with well-defined physical constraints on object points making them suitable for robust correspondence in a stereo pair of images. This is particularly useful for recovering the 3-D shape of smooth featureless surfaces from stereo which has previously been very difficult to perform.

3.1 Photometric Ratios As An Invariant

Combining equations 1, 2 and 3 gives us an expression which precisely relates pixel gray value, I , to diffuse reflection as a function of imaging geometry and camera parameters.

For incident radiance L through a small solid angle, $d\omega$, at an angle of incidence, ψ , the gray value formed from viewing angle, ϕ (assuming we subtract out the dark reference, d) is:

$$I = g [(\pi/4)(D/i)^2 \cos^4 \alpha L \rho [1 - F(\psi, n)] \times \cos \psi \times [1 - F(\sin^{-1}(\frac{\sin \phi}{n}), 1/n)] d\omega]^{1/\gamma}$$

For a general incident radiance distribution, $L(\psi)$, on an object point the gray value will be:

$$I = g [(\pi/4)(D/i)^2 \cos^4 \alpha \int L(\psi) \rho [1 - F(\psi, n)] \times \cos \psi \times [1 - F(\sin^{-1}(\frac{\sin \phi}{n}), 1/n)] d\omega]^{1/\gamma} \\ = g [(\pi/4)(D/i)^2 \cos^4 \alpha [1 - F(\sin^{-1}(\frac{\sin \phi}{n}), 1/n)] \rho]^{1/\gamma} \\ \times [\int L(\psi) [1 - F(\psi, n)] \cos \psi d\omega]^{1/\gamma} \quad (4)$$

Consider this object point first illuminated with an incident radiance distribution, $L_1(\psi)$, and then illuminated with an incident radiance distribution, $L_2(\psi)$. From equation 4 the photometric ratio of gray values is:

$$\frac{I_1}{I_2} = \frac{[\int L_1(\psi) [1 - F(\psi, n)] \cos \psi d\omega]^{1/\gamma}}{[\int L_2(\psi) [1 - F(\psi, n)] \cos \psi d\omega]^{1/\gamma}} \quad (5)$$

This photometric ratio is invariant to all camera parameters (except γ), viewing angle, ϕ , and diffuse surface albedo, ρ . Note that expression 3 for diffuse reflection is a separable function with respect to both variables, ψ , and, ϕ , and that is why the viewing angle cancels out in the photometric ratio.

3.2 Corresponding Photometric Ratios

Most cameras have a default setting of linear response (i.e., $\gamma=1.0$). If not, the intensity values can be linearized by inverse γ -correction. If gray values are linear- or are linearized- for a stereo pair of cameras, then the invariance of equation 5 guarantees that diffuse reflection from an object point will have the same ratio I_1/I_2 with respect to both cameras. Specular reflection observed from any of the stereo pair of cameras at the object point will perturb this invariance, but fortunately only diffuse reflection exists at most object points on dielectric surfaces. The numerical value of I_1/I_2 can be corresponded along epipolar lines in a stereo pair of cameras to subpixel accuracy using interpolation. This is done completely independent of knowledge of the illumination distributions $L_1(\psi)$ and $L_2(\psi)$. For a smooth surface this produces a very dense depth map from stereo correspondence. It is possible that multiple points along an epipolar line can have the same associated photometric ratio and correspondence can be aided by estimates of stereo disparity and disparity gradient [1], [14].

With 3 different illumination conditions 2 unique ratio images can be generated and now 2 photometric ratios will be invariant for object points viewed between a stereo pair of cameras, which can be used to further disambiguate object points. More than 3 different illumination conditions may provide redundant information. We intend to study the case of using more than 2 multiple illumination conditions for stereo correspondence.

3.3 Isoratio Curves and Physical Constraints

The ratio of equation 5 expresses a physical constraint consisting of the inter-relationship between the local surface orientation at an object point and the two illumination distributions, $L_1(\psi)$, and, $L_2(\psi)$. Object points having the same photometric ratios form equivalence classes that we term in this paper *isoratio curves*. Different than for *isophotes* which are curves of equal gray value in an image, an object point belongs to an isoratio curve based on the geometric relationship of its surface normal with respect to illumination independent of diffuse surface albedo. Corresponding photometric ratios along epipolar lines between a stereo pair of images is identical to corresponding points that are at the intersection of equivalent isoratio curves and the epipolar lines.

We analyze the physical constraints governing isoratio curves for distant point light source illumination. To simplify analysis somewhat we consider a Lambertian reflecting object. Consider point light source illumination at incident orientation in gradient space coordinates (p_1, q_1) , and (p_2, q_2) . These produce the following reflectance maps in gradient space respectively [6];

$$R_1(p, q) = \frac{1 + p_1 p + q_1 q}{\sqrt{1 + p^2 + q^2} \sqrt{1 + p_1^2 + q_1^2}},$$

$$R_2(p, q) = \frac{1 + p_2 p + q_2 q}{\sqrt{1 + p^2 + q^2} \sqrt{1 + p_2^2 + q_2^2}}.$$

The surface orientation of object points with photometric ratio, I_1/I_2 , is constrained in gradient space by the expression

$$\frac{I_1}{I_2} = \frac{R_1(p, q)}{R_2(p, q)} = \frac{1 + p_1 p + q_1 q}{1 + p_2 p + q_2 q} \times \frac{\sqrt{1 + p_2^2 + q_2^2}}{\sqrt{1 + p_1^2 + q_1^2}},$$

resulting in the following linear constraint in p and q :

$$(p_1 - k \frac{I_1}{I_2} p_2)p + (q_1 - k \frac{I_1}{I_2} q_2)q + 1 - k \frac{I_1}{I_2} = 0, \quad (6)$$

where

$$k = \frac{\sqrt{1 + p_1^2 + q_1^2}}{\sqrt{1 + p_2^2 + q_2^2}}.$$

Therefore object points lying on a particular isoratio curve produced from two distant point light sources all have local surface orientation that lies somewhere along this line in gradient space. Diffuse reflection resulting from expression 3 introduces a slight nonlinearity in this physical constraint. Finite light source distance also introduces nonlinearity.

While distant light source illumination is a simple case, analysis such as this provides important intuition about the correspondence of photometric ratios. For best correspondence we would like isoratio curves to intersect epipolar lines as perpendicularly as possible. We observe for equation 6 that incident light source orientations lying along, $q = mp$, the line through the origin with slope m in gradient space, produce photometric ratios with isoratio curve linear constraints that are parallel to the line $q = -(1/m)p$. These linear constraints are perpendicular to the line in gradient space along which the incident light source orientations lie. Hence, assuming nonverged cameras, there is the best chance that isoratio curves will be most perpendicular to epipolar lines if the incident directions of the light sources intersect the baseline for the stereo pair of cameras, or at least the separation between the light sources be parallel to this baseline.

4 Experimental Results

We tested the accuracy of a dense depth map determined from correspondence of photometric ratios between a stereo pair of images on a cylinder and a sphere of known ground truth. A pair of Sony XC-77 cameras with 25 mm lenses were used with a stereo baseline of 3 inches. The cameras were not verged so that the epipolar lines were the scanlines themselves. The radius of the smooth plastic cylinder shown in Figure 4 is precisely 1 3/8 inches, and the radius of the smooth sphere shown in Figure 8 is a precisely 1 3/16 inch radius billiard cue ball. The most frontal portion of these objects were placed 20 inches from the stereo baseline which is far relative to the sizes of these objects themselves. Each illumination condition was produced from one of 2 point light sources incident at approximately $\pm 25^\circ$ separated along the baseline. According to the analysis of the previous section separating point light sources parallel to the baseline increases the perpendicularity of isoratio lines with respect to epipolar lines. Again, it is not necessary at all to know how these light sources are positioned.

Figure 5 shows the photometric ratios that were used to correspond pixels for the cylinder across one scanline of the left and right stereo images. Note that there are a couple of photometric ratios on the left hand side of the left image scanline that are higher than any of the photometric ratios along the same scanline in the right image. This is because the left camera sees a small portion of the cylinder which is not in the view of the right camera. The photometric ratio of a pixel in the left image is corresponded to subpixel accuracy in the right image using linear interpolation of photometric ratios between pixels. Figure 6 shows a color bitmap of isoratio curves across the cylinder for different photometric ratios in the left and right images. Figure 7a shows the height map of the cylinder from the ground truth depth map of the cylinder and Figure 7b shows the height map of the cylinder from the derived depth map from stereo correspondence of photometric ratios. At a distance of 20 inches the average depth error across the cylinder, before smoothing, was ± 0.17 inches which is $\pm 0.85\%$.

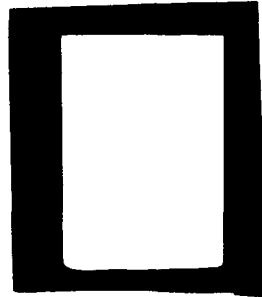


FIGURE 4

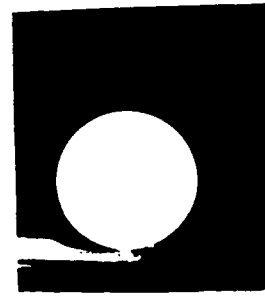


FIGURE 8

Figure 9 shows the isoratio curves for the sphere for different photometric ratios in the left and right images. Figure 10a shows the height map of the sphere from the ground truth depth map of the sphere and Figure 10b shows the height map from the derived depth map from stereo correspondence of photometric ratios. The average depth error across the sphere, before smoothing, was ± 0.09 inches which is $\pm 0.45\%$ depth variation at 20 inches.

Clearly this methodology using photometric ratios for stereo correspondence can be very accurate. While this methodology does not require any knowledge of each illumination condition, there are illumination conditions that produce better measurement accuracy than others.

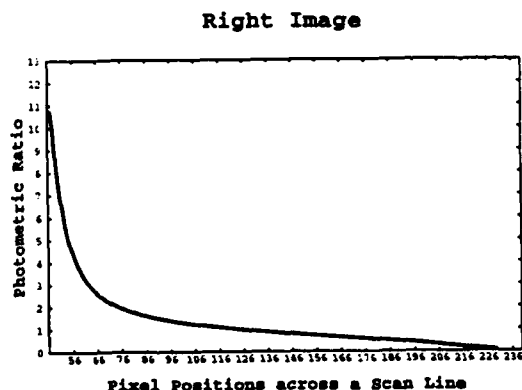
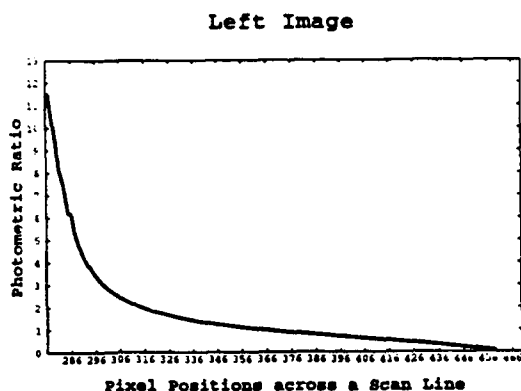


FIGURE 5

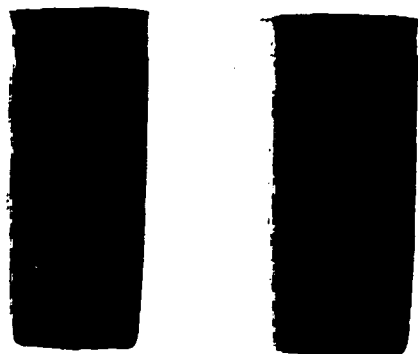


FIGURE 6

Measurement accuracy generally improves with more varying illumination conditions. With a point light source angular separation of about $\pm 10^\circ$ we were only able to get about $\pm 2\%$ average relative depth error for the cylinder and the sphere. The tradeoff of course is that with less varying illumination there is more mutually illuminated surface area. Measurement accuracy did not increase significantly for a point source angular separation beyond $\pm 30^\circ$. Higher camera signal-to-noise ratio would surely improve measurement accuracy.

We also measured depth maps for the cylinder and the sphere from correspondence of photometric ratios with light source separation vertical to the stereo baseline which according to the analysis of the previous section is least optimal. Using an angular separation of about $\pm 30^\circ$ we were still able to achieve approximately $\pm 3\%$ relative depth accuracy for both the cylinder and the sphere, which is not at all bad!

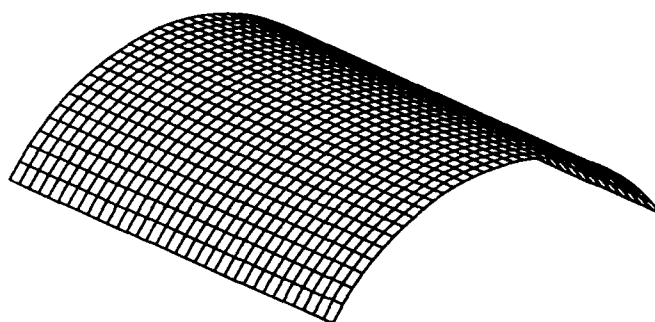


FIGURE 7a

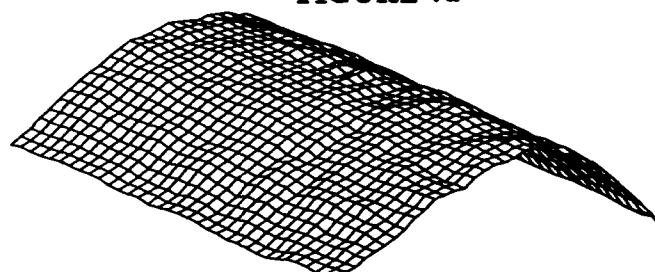


FIGURE 7b

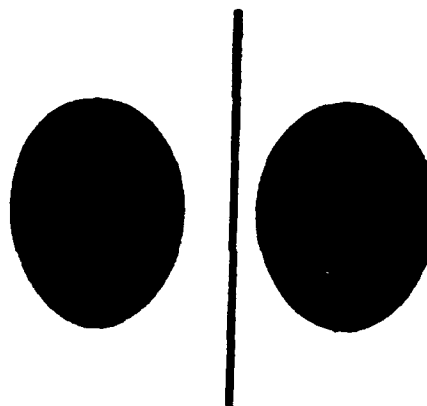


FIGURE 9

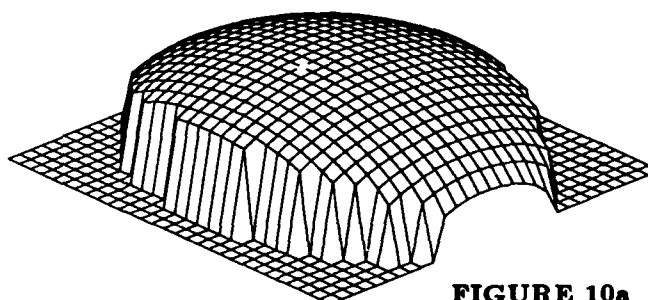


FIGURE 10a

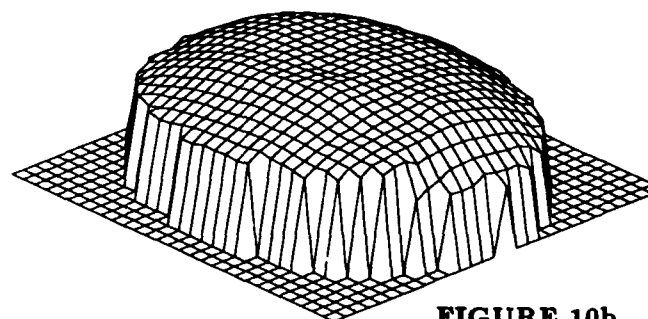


FIGURE 10b

5 Conclusion and Future Work

We have proposed and demonstrated a new practical methodology for achieving accurate correspondence of photometric values between a stereo pair of images. The advantage of using photometric values for stereo correspondence is the ability to generate dense depth maps, as well as determining the shape of smooth featureless objects. Because of characteristics that vary from camera to camera and the dependence of diffuse reflection on viewpoint, pixel gray values by themselves are unreliable for stereo correspondence of object points. We have introduced the notion of using photometric ratios, produced from different illuminations of a scene, for stereo correspondence which are reliable because of their invariance to various camera characteristics and viewpoint. Using only 2 illumination conditions we have demonstrated the high accuracy of depth determination from the stereo correspondence of photometric ratios on objects of precisely known ground truth. Because this methodology does not require knowledge of illumination conditions, and works well in perspective views, it can be applied both in machine vision and in the less controlled environments of computer vision. For instance, depth in an outdoor scene can probably be accurately determined from a stereo pair of images taken at different times with illumination varying due to position of the sun and/or variation in cloud cover. Future work will entail using this stereo methodology in these types of settings.

In addition to proposing photometric ratios as a reliable way of corresponding a stereo pair of images, we introduced the notion of the *isoratio curve* which unlike isophotes are invariant to diffuse surface albedo. Therefore isoratio curves are more directly related to the actual geometry of the surface itself and can give a large amount of information to object recognition with minimal knowledge of illumination. We are currently studying the use of isoratio curves for improving the performance of object recognition in robotic environments.

Acknowledgements

This research was supported in part by an NSF Research Initiation Award, grant IRI-9111973 and DARPA contract F30602-92-C-0191.

References

- [1] N. Ayache. *Artificial Vision for Mobile Robots: Stereo Vision and Multisensory Perception*. MIT Press, 1989.
- [2] A. Blake. Specular stereo. In *Proceedings of IJCAI*, pages 973-976, 1985.
- [3] G.J. Brelstaff and A. Blake. Detecting specular reflections using lambertian constraints. In *Proceedings of the IEEE Second International Conference on Computer Vision (ICCV)*, pages 297-302, Tampa, Florida, December 1988.
- [4] W.E.L. Grimson. *From Images to Surfaces: A Computational Study of the Human Early Visual System*. MIT Press, 1981.
- [5] W.E.L. Grimson. Binocular shading and visual surface reconstruction. *Computer Vision Graphics and Image Processing*, 28(1):19-43, 1984.
- [6] B.K.P. Horn. Understanding image intensities. *Artificial Intelligence*, pages 1-31, 1977.
- [7] B.K.P. Horn and R.W. Sjöberg. Calculating the reflectance map. *Applied Optics*, 18(11):1770-1779, June 1979.
- [8] K. Ikeuchi. Determining a depth map using a dual photometric stereo. *International Journal of Robotics Research*, 6(1):15-31, 1987.
- [9] J. H. Lambert. *Photometria sive de mensura de gratibus luminis, colorum et umbrae*. Augsburg, Germany: Eberhard Klett, 1760.
- [10] D. Marr. *Vision*. Freeman, San Francisco, 1982.
- [11] D. Marr and T. Poggio. A theory of human stereo vision. *Proceedings of the Royal Society of London B*, 204:301-328, 1979.
- [12] J.E.W. Mayhew and J.P. Frisby. *3D Model Recognition from Stereoscopic Cues*. MIT Press, 1991.
- [13] B.T. Phong. Illumination for computer generated images. *Communications of the ACM*, 18(6):311-317, June 1975.
- [14] S. Pollard, J. Mayhew, and J. Frisby. Pmf: a stereo correspondence algorithm using the disparity gradient limit. *Perception*, 14:449-470, 1985.
- [15] R. Siegal and J.R. Howell. *Thermal Radiation Heat Transfer*. McGraw-Hill, 1981.
- [16] G.B. Smith. Stereo integral equation. In *Proceedings of the AAAI*, pages 689-694, 1986.
- [17] D. Terzopoulos. The role of constraints and discontinuities in visible-surface reconstruction. In *Proceedings of IJCAI*, pages 1073-1077, 1983.
- [18] L.B. Wolff. *A Diffuse Reflectance Model for Dielectric Surfaces*. Johns Hopkins Technical Report CS-92-04, April 1992 (Submitted to the Journal of the Optical Society of America).
- [19] L.B. Wolff. Diffuse and specular reflection. *Proceedings of the DARPA Image Understanding Workshop (These Proceedings)*, April 1993.
- [20] R.J. Woodham. *Reflectance map techniques for analyzing surface defects in metal castings*. PhD thesis, MIT AI Lab Tech Report AI-TR-457, June 1978.
- [21] A. Zisserman, P. Giblin, and A. Blake. The information available to a moving observer from specularities. *Image and Vision Computing*, 7(1):38-42, 1989.

Learning and Feature Selection in Stereo Matching

Michael S. Lew Thomas S. Huang Kam W. Wong
University of Illinois at Urbana-Champaign

Abstract

We present a novel stereo matching algorithm which integrates learning, feature selection, and surface reconstruction. First, an instance based learning (IBL) algorithm is used to generate an approximation to the optimal feature set for matching. Second, we develop an adaptive method for refining the feature set. This adaptive method analyzes the feature error to locate the sources of mismatches. Then the search through feature space for maximizing the class separation function is guided by eliminating the sources of mismatches. Third, we introduce a method for determining when apriori knowledge is necessary for discriminating between the correct match and the sources of mismatches. If the apriori knowledge is necessary then we use a surface reconstruction model to discriminate between match possibilities. We performed comprehensive comparison of our algorithm and a traditional pyramid algorithm on a wide set of real images. Finally, extensive empirical results of our algorithm based on real images are presented.

1 Introduction

Stereo matching is an important problem in computer vision. It is necessary for passive range finding. It greatly simplifies navigation and automated modeling of objects and terrains. In human body measurement, we could automatically create models of human bodies for manufacturing apparel or create ergonomic equipment for the office or home use. In flight simulation and the new area of virtual reality, we could automatically create models of terrain and other natural environments.

In the final report of the NSF Workshop on "Challenges in Computer Vision Research; Future Research Directions," two of the major recommendations on research topics and issues included [S. Negahdaripour and A. Jain 1992].

- More experimental rigor in vision research
- Researchers should address the integration of isolated modules at each visual processing level

This paper addresses these two issues by performing extensive empirical testing of our algorithm on a wide range of real images, and integrating the modules of learning, feature selection, and surface reconstruction.

We will use the following assumptions and definitions:

We are given two intensity images, L and R.

(x_l, y_l) denotes the image axes of L

(x_r, y_r) denotes the image axes of R

(x_p, y_p) denotes a specific point in (x_l, y_l)

(x_c, y_c) denotes a specific point in (x_r, y_r)

Correspondence denotes a list of two elements, (x_p, y_p) and (x_c, y_c) .

We are given n feature classes (e.g. intensity, Laplacian, etc.).

Feature set refers to a set whose members are feature classes

Feature vector refers to a vector of size n composed of the values of every feature class at a specific point.

In stereo matching, our goal is to find correspondences between two intensity images of roughly the same content. Given knowledge of the camera calibration and the correspondence (x_p, y_p) to (x_c, y_c) we can then reconstruct the 3-D coordinates of the object in the world.

This paper presents a new multi-feature stereo matching algorithm. We use the following features in our algorithm:

- image intensity
- x derivative of intensity
- y derivative of intensity
- gradient magnitude
- gradient orientation

- Laplacian of intensity
- curvature of 2D edges

In the sense of using "Landmark" as something which is unique, we call our algorithm the "Landmark Stereo Matching Algorithm" because the central idea of the method is to find a feature set for (x_p, y_p) that will make the point unique.

There are essentially three steps in our method which is shown in Figure 1. The first step produces an approximation toward the optimal feature set. The second step refines the feature set toward the optimal feature set in the sense of making the selected point (x_p, y_p) unique. The third step treats the case where the feature set is ambiguous.

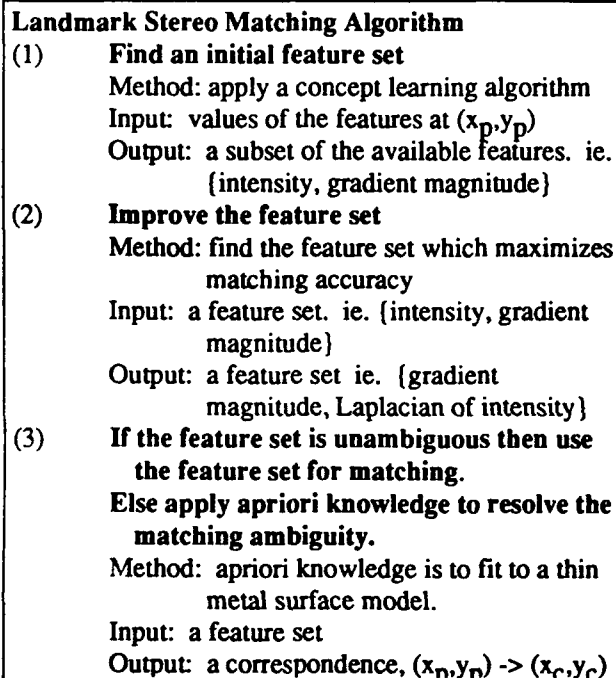


Figure 1. The Landmark stereo matching algorithm.

The history of stereo research has provided a rich and extensive background for the ideas in our research. Moravec [1980] found interesting points in the left image and used the binary-search method of an image pyramid to find the correspondences. Hannah made improvements to his method but kept the unidirectional coarse-to-fine search method [M. J. Hannah 1988]. Marr, Poggio, and Grimson used zero-crossings of the Laplacian of Gaussian at different spacings as matching primitives. They found matches at a particular initial level, enforced continuity of zero-crossings, and then approximated the match down the pyramid from coarse-to-fine [D. Marr and T. Poggio 1980; W. E. L. Grimson 1981]. Lim and Binford [1987] used a

hierarchical structure based on different scale features, specifically, bodies, surfaces, junctions, curves, and edges. Hoff and Ahuja [1985] used zero-crossings to integrate surface modeling and stereo matching at a particular initial level and strictly approximated toward the finest level. Surfaces were modeled as planar and quadratic patches. Barnard [1987] used an annealing approach for finding global optima from matching all points simultaneously. The match error was an energy function combining intensity difference and local changes in disparity. Cohen, Vinet, and Sander [1989] used an edge hierarchy to integrate segmentation with stereo matching.

The progression of stereo research appears to be toward using more features of varying levels of abstraction. The most recent work includes using a variety of waveforms as primitives [McKeown and Hsieh 1992]. Marapane and Trivedi [1992] used multiple primitives in a hierarchy [Marapane and Trivedi 1992]. In addition, Weng, Ahuja, and Huang [1992] used edgeness, and positive and negative comerness in a hierarchical based matcher.

This paper is organized as follows. Section 2 describes the instance based learning algorithm, which is essentially step 1 of our algorithm. Section 3 describes the feature set improvement, which corresponds to step 2 of our algorithm. Section 4 describes the actual stereo matching once the feature set has been found. It also explains the method of resolving ambiguous matches, which is step 3. Section 5 presents the results of using the algorithm upon the set of real images. Section 6 summarizes the conclusions and contributions.

2 Instance Based Learning

This section explores the problem of finding an initial feature set in step 1 of Figure 1. Given that the combinatorial explosion from searching for an optimal feature set may be prohibitive, we explore a method of finding an initial point from which to begin the search through feature space. Computational expense can be saved by generating a first approximation of the optimal feature set using a concept learning algorithm such as a neural net or an instance-based learning algorithm [Aha and Kibler 1989].

Some preliminary terminology for instance based learning algorithms is reviewed below:

Exemplar refers to a list of two elements, where the first element is a feature vector, and the second element is a feature set. The

classification of the feature vector is assumed to be the associated feature set.

Exemplar list refers to a list of exemplars.

There are two stages in the instance based learning paradigm. First, a training set which has the form of an exemplar list is used to build a concept, *C*. This is called the learning stage. Second, after the training set has been fully processed, new input feature vectors are classified using *C*. A simplistic IBL algorithm would simply copy the training set to *C* for the learning stage. A simplistic IBL algorithm classifies input feature vectors as follows:

- (1) Find the euclidean distance between the input feature vector and the feature vector of each exemplar in *C*.
- (2) Classify the input vector as the feature set of the vector in *C* which has the minimum distance.

Advantageous characteristics of instance-based learning algorithms [Aha and Kibler 1989] are (1) simple representations for concept descriptions, (2) low incremental learning costs, (3) small storage requirements, (4) produce concept exemplars on demand, (5) ability to learn continuous functions, and (6) ability to learn non-linearly separable categories.

Consider an example where the input feature vector for (*x_p*, *y_p*) is [100,9,20]. The feature vector of the exemplar on line (2) of Figure 2 is closest to the input feature vector. Then, we would classify the input feature vector as {intensity}.

Line	Concept C
(1)	([100, 0, 20], {intensity})
(2)	([100, 10, 20], {intensity})
(3)	([100, 30, 20], {orientation})
(4)	([100, 40, 20], {orientation})

Figure 2. An example of a concept containing 4 exemplars. The feature classes of the feature vector are [intensity, magnitude, orientation]. Intensity and orientation are the only possible classifications.

Our approach toward noise tolerance is to classify the new element using the entire concept *C* instead of the nearest neighbor. Specifically, we accumulate the support from each exemplar in *C* toward a particular classification. The classification with the highest support is then chosen. The Gaussian weighted support function is chosen as

$$S(k) = \sum_{i \in C, k} e^{-\frac{1}{2} \left(\frac{\delta(f_i, f_r)}{\sigma} \right)^2}$$

where $\delta(f_i, f_r)$ is the metric between the exemplars in *C* which are in class, *k*, and the new instance, *f_T*. Then the classification for *f_T* would satisfy

$$\max(S(k)), k = 1..c$$

where *c* is the number of classes. Intuitively, this will result in a few incorrect points being suppressed by the vote of the many correct points, as the Gaussian weighting will give greater support to nearby exemplars, and less support from farther exemplars in feature space.

Another issue in designing instance based learning algorithms is minimizing the size of the concept *C*. Note that in Figure 2, the classification for any new input feature vector will not change if we eliminate the exemplars on lines 1 and 4. In general, we can reduce the size of *C* by grouping exemplars, which are close in feature space, into a single exemplar. Furthermore, if the feature vector of an exemplar in *C* has a different Gaussian support classification than it's associated classification, then we can delete the exemplar from *C*.

The instance-based learning algorithm which is used for the Landmark algorithm is based upon Growth NT[Aha and Kibler 1989] with some modifications toward improving training order independence (creating the discard list), and concept size minimization. Given that *T* denotes the training set, NT2 is shown in Figure 3.

Instance Based Learning Algorithm

- (1) Initialize *C* to the set of first exemplar in *T*.
- (2) For all subsequent training exemplars *t* in *T*:
- (3) *k* = Gaussian Support Classification of *t* by *C*.
- (4) If (*k* equals the associated classification of *t*) THEN add *t* to the discard list ELSE add *t* to *C* and check the discard list for incorrectly classified instances.
- (5) Delete redundant and noisy exemplars from *C*.

Figure 3. NT2, the instance based learning algorithm. This algorithm shows the first stage of instance based learning algorithms, where the concept is created.

3 Feature Set Improvement

This section explores the problem of improving the current feature set in step 2 of Figure 1. In order to optimize the feature set, we need a function to maximize. We shall use the concept of the class separation distance in formulating the optimality criterion. Consider that each feature set will result in a

specific error function with respect to (x_p, y_p) . The optimal feature set should have the quality that its minimum is at the correct (x_c, y_c) . The ideal error function would have one minimum at the correct (x_r, y_r) , whereas the worst case error function would be a flat plane, which would be the most ambiguous situation.

By defining the class separation distance in terms of the error function, we only need to consider the minima instead of every point in the image. This allows us to significantly reduce computational expense. Let us consider the case of multiple minima in the error function. Each minimum is associated with a different stereo correspondence, where only one correspondence is correct. The other minima are then called sources of mismatches since they lead to incorrect correspondences.

With respect to stereo matching, we want to maximize the distance between the value of the error function at the correct correspondence and the value of the error function at all the sources of mismatches.

Thus, we define a measure which will give a higher class separation distance to greater differences between minima in the error function, that is, when the difference between the first and second minima is large, the class separation distance is also large. But, we would also like the class separation distance to gracefully diminish with additional minima which are close to the global minimum. The Gaussian function was chosen because of these properties.

Let n be the total number of available features, then the error function for a feature set is

$$\text{Error} = \mathbf{w} \cdot \mathbf{F}$$

where

$$\mathbf{w} = [w_1 \ w_2 \ \dots \ w_n]$$

with

$$\sum_{i=1}^n w_i = 1$$

and

$$\mathbf{F} = [f_1 \ f_2 \ \dots \ f_n]$$

where f_n is the error due to the n th feature with respect to (x_p, y_p) . Note that f_1 refers to $f_1(x_r, y_r)$, which is the error from using the first feature class between (x_p, y_p) and every point in (x_r, y_r) . Similarly, f_2 refers to $f_2(x_r, y_r)$, which is the error from using the second feature class.

Then if we choose Gaussian weighting, the class separation distance becomes

$$J(\mathbf{w} \cdot \mathbf{F}) = 1 - \frac{1}{N_v} \sum_{\mathbf{x} \in M} e^{\frac{1}{2} \left(\frac{\mathbf{w} \cdot \mathbf{F}(\mathbf{x}_G) - \mathbf{w} \cdot \mathbf{F}(\mathbf{x})}{\sigma} \right)^2}$$

where \mathbf{x}_G refers to the global minimum of the error function, N_v refers to the total number of minima, and M represents all \mathbf{x} such that \mathbf{x} is a local minimum but not \mathbf{x}_G . Thus the class separation, J varies between 0 and 1, the minimum and maximum class separation distances, respectively.

In image space, sufficient conditions for a local minimum are

$$(\mathbf{w} \cdot \mathbf{F}(\mathbf{x}))_x = 0 \text{ and } (\mathbf{w} \cdot \mathbf{F}(\mathbf{x}))_y = 0$$

and

$$\begin{vmatrix} (\mathbf{w} \cdot \mathbf{F}(\mathbf{x}))_{xx} & (\mathbf{w} \cdot \mathbf{F}(\mathbf{x}))_{xy} \\ (\mathbf{w} \cdot \mathbf{F}(\mathbf{x}))_{yx} & (\mathbf{w} \cdot \mathbf{F}(\mathbf{x}))_{yy} \end{vmatrix} > 0$$

Now, the goal is to find \mathbf{w} such that J is maximized, or

$$J(X) = \max_{\mathbf{w}} J(\mathbf{w} \cdot \mathbf{F})$$

With consideration of all possible feature sets, we have reached the discrimination limit of the feature classes and metrics. For suboptimal search we could stop searching at a sufficiently high class separation. Henceforth, this will be called a distinct feature set as opposed to an optimal feature set. Nevertheless, it is possible that there is no \mathbf{w} which results in a single distinct or optimal minimum. This case is explored in section 4.

A straightforward method of feature selection is to maximize $J(\cdot)$ between (x_p, y_p) and R . Intuitively, this will result in obtaining the most distinct error function from the set of feature classes. If we were to use this approach, we would also use one of the traditional feature selection search methods: Branch and Bound [Narendra and Fukunaga 1977] for the optimal feature set, or one of many feature selection algorithms [Whitney 1971; Kittler 1978; Marill and Green 1963; Kittler 1978] for a less computationally expensive but suboptimal set. We have another interesting possibility.

Instead of maximizing $J(\cdot)$ between (x_p, y_p) and R , we could maximize $J(\cdot)$ between (x_p, y_p) and L . This possibility has the following significant advantages: (1) we would know which minimum in the error function corresponds to (x_p, y_p) and which minima correspond to sources of mismatches; (2) if we compute the error only at the minima for the features not in the feature set, then we could guide the addition of features to the

feature set by adding the feature which has the greatest total error at the minima.

The disadvantage is that once the feature set is selected, we will have to search through R for the global minimum, which in the straightforward method would already have been performed.

Both methods share the important advantage of the ability to determine when the feature set is insufficient for matching (x_p, y_p) . This situation occurs when the class separation distance is lower than a threshold, J_t . Although many stereo matchers will reject a correspondence if the final feature error is too high, it is rare for a stereo matching algorithm to be able to determine if there are too many points with low feature errors (ie. when $J < J_t$).

Let F_c = list of n feature classes. The algorithm is shown in Figure 4.

Feature Set Improvement Algorithm

- (1) F_i = Feature set approximation from the IBL algorithm with respect to (x_p, y_p)
- (1.1) $J_{old} = J(F_i)$ = Initial class separation
- (2) if the feature set is distinct ($J(F_i) > J_t$) then go to Stereo Matching Algorithm (in Section 4 or step 3 of Figure 1.)
- (3) M_t = minima in F_i applied to L and (x_p, y_p)
- (4) Apply M_t to F_c .
- (5) Let f be the element in F_c which has the maximum total error over M_t .
- (5.1) If there are no features left (f=NULL) then go to Stereo Matching Algorithm
- (5.1) Add f to F_i .
- (5.2) Delete f from F_c
- (5.3) if the current feature set is distinct ($J(F_i) > J_t$) then go to Stereo Matching Algorithm
- (5.4) if $J(F_i) > J_{old}$ then $J_{old} = J(F_i)$; go to 5
- (5.5) Delete f from F_i . Go to 5

Figure 4. The feature set improvement algorithm.

In summary, the central idea of the guided or adaptive method of finding distinct feature sets is to record the points of minima of the current feature set with respect to (x_p, y_p) . Then, features which have the largest total error at the minima are added to the feature set if J increases. Thus, we can perform an informed addition of features to the feature set. This information constrains the searching in an intuitively pleasing manner.

4 Stereo Matching Algorithm

In step 3 of Figure 1, we have two possibilities:

- (1) The feature set, F_i , is distinct
- (2) The feature set, F_i , is not distinct

If the first is true, then the feature set appears to be able to discriminate between the correct match and the sources of mismatches. Consequently, we apply the feature set to the corresponding epipolar line of R, and determine the correspondence as the point of minimum error.

In the second case, the discriminatory ability of our feature set is insufficient to properly distinguish between the possible matches in R. There are two options in this situation. We could reject the point, or apply a heuristic to select one of the minima. Thus, the solution will depend upon the particular application to which the feature selection is being applied.

For stereo matching we chose to decide between match possibilities by fitting the previous matches and the current match to the quadratic variation, E [Grimson 1981; Terzopoulos 1983, 1984, 1988].

$$E = \iint \left(\frac{\partial^2 u}{\partial x^2} \right)^2 + \left(\frac{\partial^2 u}{\partial y^2} \right)^2 dx dy$$

The surface reconstruction method of Harris [1987] was chosen because it could potentially be implemented in hardware, and it can incorporate slope information. The quadratic variation including slope information is shown below

$$E = \iint \left[(ux - p)^2 + (uy - q)^2 + p_x^2 + p_y^2 + q_x^2 + q_y^2 \right] dx dy$$

If we consider (x_p, y_p) and the sources of mismatches as a set of points in L. Then map them to the minima in R, the interpolated depth can be used to compute the quadratic variation. Thus, the correspondence which satisfies

$$\min \{ w \cdot F + \text{Quadratic Variation} \}$$

over (x_p, y_p) and the sources of mismatches in L and the minima in R is taken as the correct correspondence of (x_p, y_p) . Furthermore, if we have access to a dense surface map generated from previous correspondences between L and R, then we can also use the surface map to fit (x_p, y_p) directly.

The stereo matching algorithm which incorporates the two cases is shown in Figure 5.

Stereo Matching Algorithm

- (1) If the feature set, F_i is distinct, then go to 2
Else go to 3
- (2) Apply F_i to R and (x_p, y_p) to find the correspondence, (x_c, y_c)
 - (2.1) Update surface reconstruction map $z=f(x, y)$
 - (2.2) Update the learning algorithm concept, C .
 - (2.3) STOP
- (3) If the reconstructed surface map is dense then set (x_c, y_c) to the minimum which agrees closest with the surface reconstruction map. If the map is not dense then select the correspondence which minimizes the normalized sum of the feature error and the quadratic variation, E over (x_p, y_p) and the sources of mismatches..
 - (3.1) Update surface reconstruction map.
 - (3.2) STOP

Figure 5. The stereo matching algorithm.

5 Results

In this section we test the Landmark algorithm and a conventional pyramidal algorithm upon the test set of real images. We present the comparative matching accuracies, and then we show the test images, and other visual representations of the matches found from the Landmark algorithm.

The algorithms were written in the C programming language, and were ported from a UNIX-based Hewlett-Packard workstation to a 50-MHX Zeos 486/AT.

The Landmark Algorithm and a pyramidal algorithm were tested on stereo pairs of real images. The results are shown in Figure 6, and graphed in Figure 7. The pyramidal algorithm is presented as a benchmark. The data structure for the pyramidal algorithm is a Gaussian intensity pyramid. Linear search was performed at the starting level, and then hill climbing through the pyramid image structure was used to implement the refinement to the finest resolution. The matching feature was intensity and the metric was the normalized correlation coefficient.

The 256x256 images include a poster of a volleyball player, a person standing in front of a plain background, an outdoor scene of a street, and a rock wall image from the Stuttgart standardized image set [Forstner, W. 1986; Gulch, E. 1988], robot arms against a complex background, and a face.

Unfortunately, due to space limitations we can not include all of the visual results in this paper.

The volleyball poster in Figure 8 was chosen because it demonstrates the basic matching accuracy in object space. Since all of the points lie on a plane it is trivial to check whether a match is correct. The greatest deviation in the Z axis for the reconstructed surface from the average was 0.8 cm. Figure 9 shows the 8216 matched points. Figure 10 shows the reconstructed surface from the matched points using the Landmark algorithm.

For the following images, the matches were checked visually. Thus the expected accuracy is approximately one pixel.

The rock wall stereo pair in Figure 11 depicts a rock wall which changes rapidly in depth. This image was selected because it shows the potential for a stereo matcher to perform automated terrain mapping, and for benchmark reasons. It was in the "difficult" category of the Stuttgart standardized image set [Forstner 1986, Gulch 1988]. Figure 12 shows the 2281 matched points. The reconstructed surface in Figure 13 shows the mismatched points as sharp jagged peaks.

The robots stereo pair in Figure 14 shows three industrial robot arms. This stereo pair was chosen for the similarity to industrial manufacturing environments. Note that points along the background, the robots, and the curved white tubing were matched. The outline of the two robots on the left can be seen against the door. The outline of the robot on the right side can be seen against the wall. Figure 15 shows the 1275 matched points. The reconstructed surface is shown in Figure 16.

Figure 17 shows the face stereo pair. This stereo pair was chosen to show the potential for human body measurement using stereo matching. Note that the resolution was sufficient to show the eyes and nose, but not the lips. The matched points occur in curves, because these represent equal intensity areas on the face. Figure 18 shows the 726 matched points. The reconstructed surface in Figure 19 shows that the eyes are slightly too sunken. This is due to the limitations of the resolution of the image to resolve depth sufficiently accurately.

Finally, we briefly consider the effect of the learning module. The learning module is used in order to reduce the time required to search for a distinct feature set, and that the time used by the learning module will depend upon the size of the exemplar list as well as the

size of the training instances. When we switched off the learning module, the average time was 1.03 sec/point. When we used an exemplar list of maximum length 100 with the seven features described in section 1, the average time was 0.44 sec/point. Thus, the learning module resulted in more than doubling the matching speed.

	Poster	Person	Street	Wall	Robots	Face
Pyramid - 16x16	78	71	48	61	67	52
Pyramid - 32x32	85	74	75	69	84	89
Pyramid - 64x64	83	84	68	74	79	95
Landmark	97	96	91	97	95	98

Figure 6. A table of the comparative matching accuracies in percentage of the Landmark algorithm versus the pyramid algorithm. Note that the pyramid algorithm required a starting level at which the initial matches were found by linear search. The numbers such as 16x16, 32x32, or 64x64 refer to the resolution of the starting level.

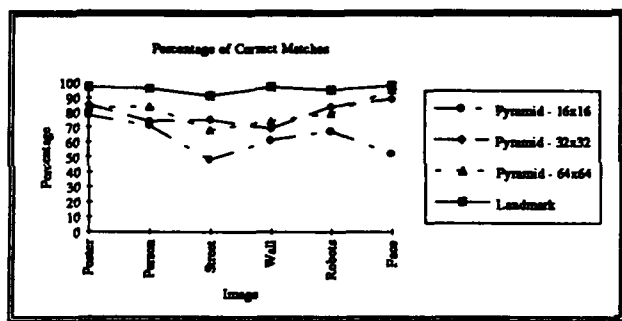


Figure 7. The percentage of correctly matched points over a variety of images between a pyramidal matcher with starting resolutions at 16x16, 32x32, and 64x64, and the Landmark algorithm.

6 Conclusions

For practical applications, a robust stereo matching algorithm should be (1) sufficiently general to analyze the image content and select the appropriate features; (2) it should use features which minimize the possibility of a wrong match; (3) it should be able to realize its own limitations. Specifically, it should know when it is unable to find a reliable and accurate match; (4) it should not require excessive computational power nor resources; (5) it should be sufficiently noise tolerant to match real world scenes as opposed to artificial or laboratory scenes.

The Landmark stereo matching algorithm satisfies all of the above properties by (1) using a noise tolerant instance based learning algorithm to generate an approximation to the optimal feature set; (2) using an adaptive method to maximize the distinctness of the selected point (x_p, y_p); (3) implementing the complete algorithm on a personal computer, (4) extensively testing the Landmark algorithm upon a wide range of real images.

Furthermore, from the final report of the NSF Workshop on "Challenges in Computer Vision Research; Future Research Directions," two of the major recommendations on research topics and issues included [S. Negahdaripour and A. Jain 1992].

- More experimental rigor in vision research
- Researchers should address the integration of isolated modules at each visual processing level

The Landmark algorithm addresses both of these recommendations.

The set of test images were real images of complex scenes which would be found in practical applications such as terrain mapping, human body measurement, and industrial manufacturing. For the purposes of bench marking, the Landmark algorithm was compared to a single feature pyramid matching algorithm. The matching accuracy of the Landmark algorithm ranged from 91% to 99% while the matching accuracy for the pyramid algorithm ranged from only 52% to 95%.

The main contributions of this paper are

- (1) Integrating the modules of learning, feature selection, and surface reconstruction.
- (2) Extensive empirical testing upon real images
- (3) A method to determine when the feature set is insufficient to discriminate between match possibilities.
- (4) A method of guided maximization of the class separation criterion for selecting the best feature set.
- (5) A noise tolerant instance based learning algorithm

Future research will focus on recognition applications, in particular, facial feature recognition.

The software for this algorithm, and a stereo workbench complete with automated testing routines, stereo test images, manually found correspondences, and graphics input and output routines is available

from the authors for the purposes of bench marking and comparison with other methods. The PC executable requires a 386 PC running Microsoft Windows version 3.1, and 8 MB of RAM.

Acknowledgments

The research reported in this paper was conducted as part of University of Illinois Advanced Construction Technology Center research program sponsored by the U.S. Army Research Office under the DoD-University Research Initiative Program, and National Science Foundation Grant IRI-89-02728.

References

- Aha, D. W., and D. Kibler, "Noise-Tolerant Instance-Based Learning Algorithms," *Proc. of the International Joint Conference on Artificial Intelligence*, pp. 794-799, 1989.
- Barnard, S. T., "Stereo Matching By Hierarchical, Microcanonical Annealing," *Proc. of Image Understanding Workshop*, February 1987.
- Cohen, L., L. Vinet, and P. Sander, "Hierarchical Region Based Stereo Matching," *Proc. of Computer Vision and Pattern Recognition*, pp. 416-421, 1989.
- Devijver, P. A., and J. Kittler, *Pattern Recognition: A Statistical Approach*. Prentice Hall, 1982.
- Forstner, W., Institute for Photogrammetry, University of Stuttgart, West Germany, 1986.
- Grimson, W.E.L., *From Images to Surfaces: A Computational Study of the Human Early Visual System*. MIT Press. Cambridge, MA, 1981.
- Gulch, E., "Results of Test on Image Matching of ISPRS WG III/4," *International Archives of Photogrammetry and Remote Sensing*, vol. 27-III, pp. 254-271, 1988.
- Harris, J. G., "A New Approach To Surface Reconstruction: The Coupled Depth/Slope Model", *First International Conference on Computer Vision*, pp. 277-283, 1987.
- Hannah, M. J., "Digital Stereo Image Matching Techniques," *ISPRS*, Kyoto, vol. 27, 1988.
- Hoff, W. A. and N. Ahuja, "Depth From Stereo," *Proc. of Fourth Scandinavian Conference on Image Analysis*, June 18-20, Trondheim, Norway, pp. 761-768, 1985.
- Kittler, J., "Une generalisation de quelques algorithmes sous-optimaux de recherche d'ensembles d'attributs," *Proc. Congres AFCET/IRIA, Reconnaissance des Formes et Traitement des Images*, Paris, pp. 678-686, 1978.
- Lim, H. S. and T. O. Binford, "Stereo Correspondence: A Hierarchical Approach," *Proc. of Image Understanding Workshop*, February, 1987.
- Marapane, S.B. and M.M. Trivedi, "Multi-Primitive Hierarchical (MPH) Stereo System," *Proc. 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 499-505, 1992.
- Marill, T., and D.M. Green, "On the Effectiveness of Receptors in Recognition Systems," *IEEE Trans. Inform. Theory*, vol. 9, pp. 11-17, Jan. 1963.
- Marr, D. and T. Poggio, "A Theory Of Edge Detection," *Proc. R. Soc. London*, vol. B 204, pp. 301-328, 1980.
- McKeown, D. M. and Y.C. Hsieh, "Hierarchical Waveform Matching: A New Feature-Based Stereo Technique", *Proc. 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 513-519, 1992.
- Moravec, H. P., "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," Ph.D. Thesis, Stanford University, Computer Science Department Report STAN-CS-80-813, September 1980.
- Narendra, P.M., and K. Fukunaga, "A Branch and Bound Algorithm for Feature Subset Selection," *IEEE Trans. Comput.*, vol. 26, pp. 917-922, Sept. 1977.
- Negahdaripour, S., and A. K. Jain, "Challenges in Computer Vision: Future Research Directions," *Proc. 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 189-199, 1992.
- Poggio, T. and C. Koch, "Ill-posed problems in early vision: from computational theory to analogue networks," *Proc. R. Soc. Lond. B* 226, 303-323, 1985.
- Terzopoulos, D., "Multi-level Computational Processes for Visual Surface Reconstruction", *Computer Vision, Graphics, and Image Processing* 24, pp. 52-96, 1983.
- Terzopoulos, D., "Multiresolution Computation of Visible-surface Representations", Ph.D. Thesis, MIT, Cambridge, MA, 1984.
- Terzopoulos, D., "The Computation of Visible-Surface Representations", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 4, pp. 417-437, 1988.
- Weng, J., N. Ahuja and T. S. Huang, "Matching Two Perspective Views," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 8, pp. 806-825, 1992.
- Whitney, A., "A Direct Method of Nonparametric Measurement Selection", *IEEE Trans. Comput.*, vol. 20, pp. 1100-1103, 1971.

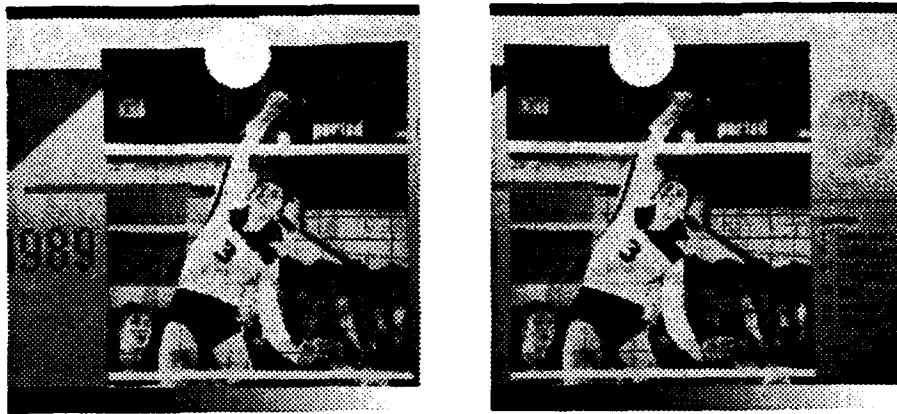


Figure 8. Volleyball poster stereo pair

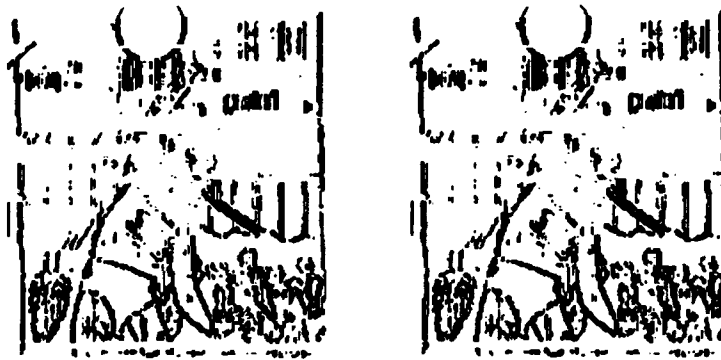


Figure 9. Matched points of the volleyball stereo pair

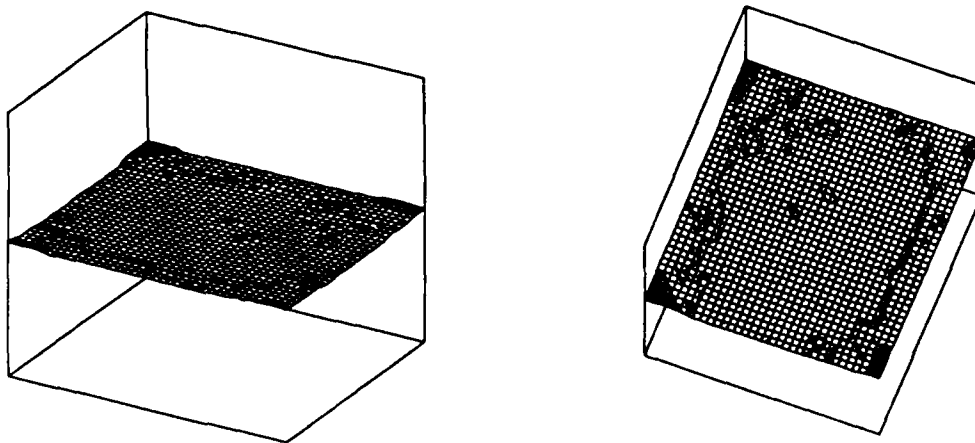


Figure 10. Reconstructed surface of the volleyball poster stereo pair at different viewing angles

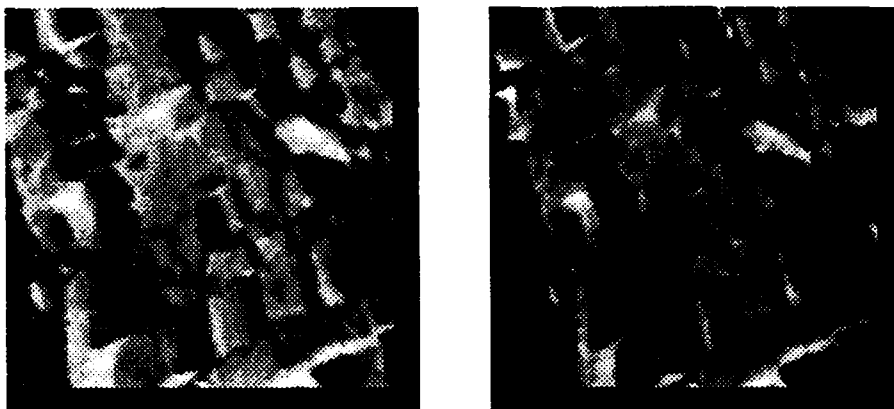


Figure 11. Rock wall stereo pair



Figure 12. The Matched Points of the Rock Wall Stereo Pair

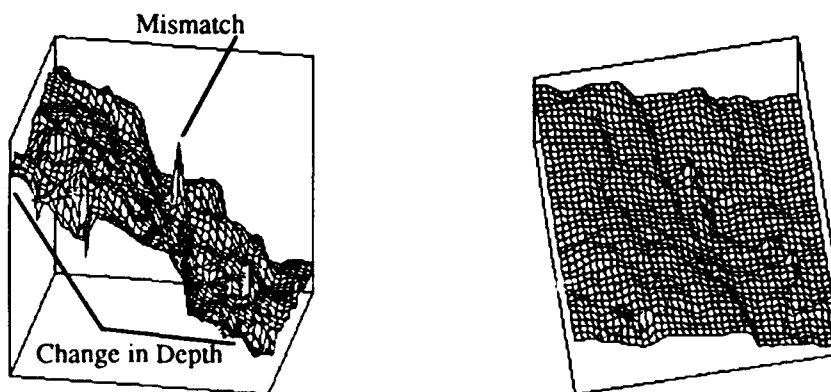


Figure 13. Reconstructed surface of the rock wall stereo pair at different viewing angles. Note that the rock wall is a slanted wall which slants away from the viewer. The spike in the left surface plot is due to a mismatch.



Figure 14. Robots stereo pair



Figure 15. The matched points of the robots stereo pair

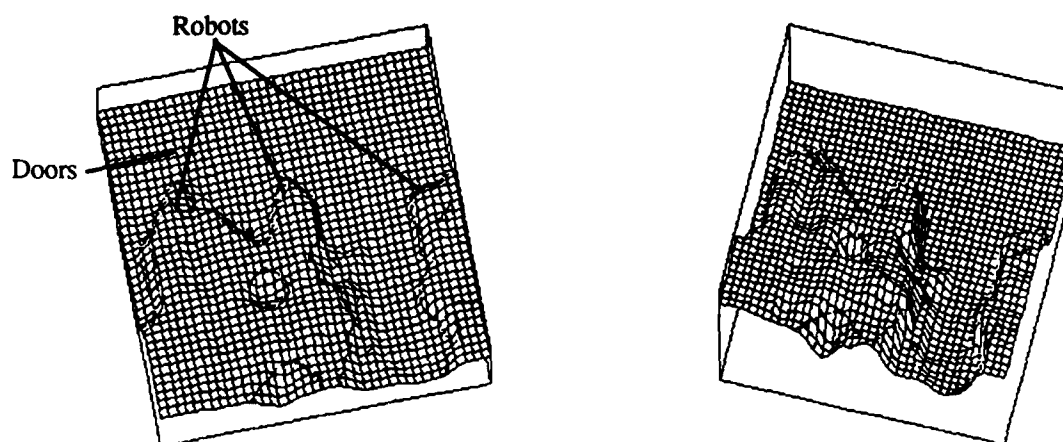


Figure 16. Reconstructed surface of the robots stereo pair at different viewing angles

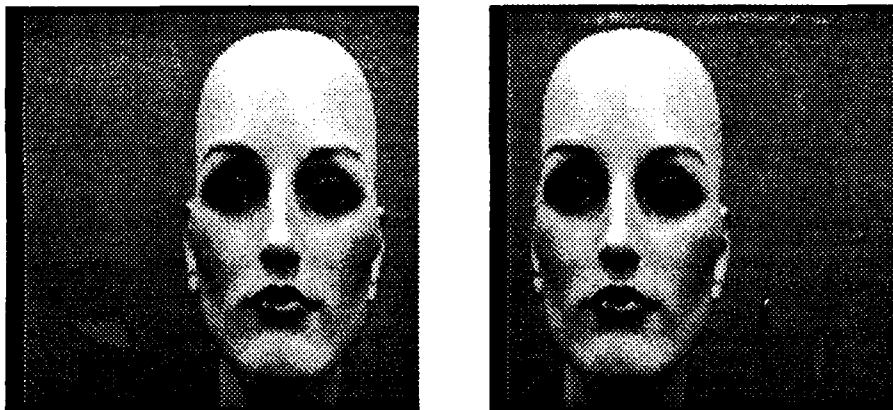


Figure 17. The face stereo pair

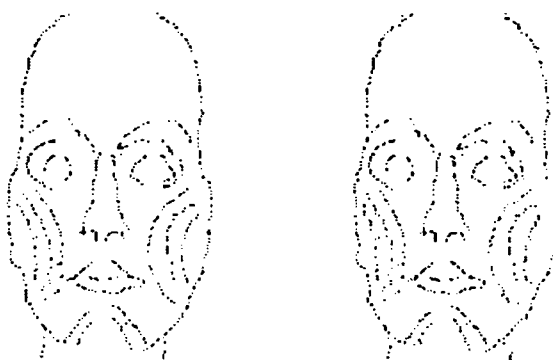


Figure 18. Matched points of the face stereo pair

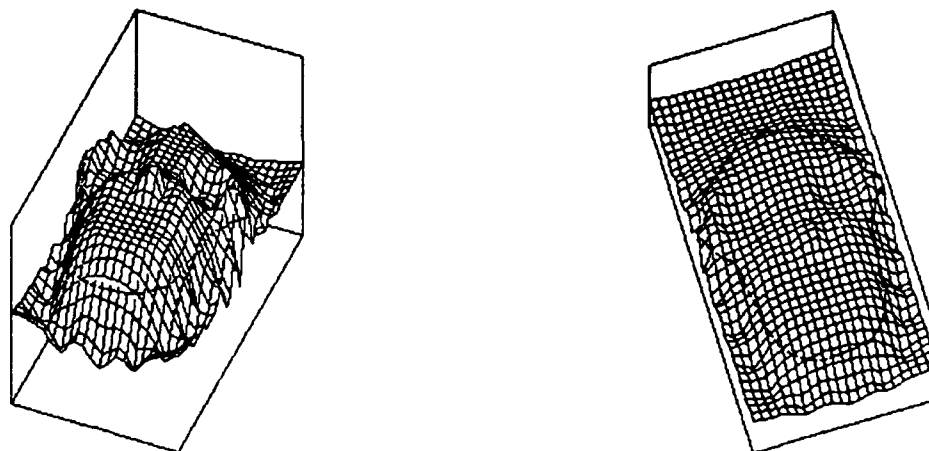


Figure 19. Reconstructed surface of the face stereo pair at different viewing angles

Implementation and Performance of Fast Parallel Multi-Baseline Stereo Vision

Jon A. Webb

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

A fast implementation of multi-baseline stereo vision on a parallel computer is described. For three 240×256 images, the algorithm runs in 64 ms on 64 iWarp processors, exceeding 15 Hz frame rate. This is a speedup of 51 over an implementation on a SPARC II and represents the fastest correlation-based stereo vision system reported. Implementing this algorithm this efficiently required careful trade-offs in algorithm design and, particularly, in the implementation of the basic communication operations. A building block approach is described for achieving best efficiency in communication; the basic operations that the parallel computer can do at maximum speed are identified, and then these primitives are used to construct the communications functions needed by the algorithm.

1. Introduction

We have achieved the highest reported performance ever of a stereo vision system based on correlation: 64 ms for 240×256 imagery, exceeding 15 Hz on a 64-processor iWarp computer. Achieving this was not simply a matter of increasing processor performance but involved

This research was supported by the National Science Foundation under Grant MIPS 8920420 and by the Defense Advanced Research Projects Agency and monitored by U. S. Army Training and Doctrine in Fort Huachuca, Arizona under Contract DABT63-91-C-0035.

The Government has certain rights in this material. Any views, opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation. They should also not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

many careful trade-offs that apply to vision systems and parallel machines in general. This paper closely analyzes the performance of this one system in order to derive general lessons concerning the performance of parallel systems on computer vision problems.

2. Stereo vision algorithm

The stereo vision algorithm used here is Kanade-Okutomi multi-baseline stereo [2]. Multi-baseline stereo uses multiple cameras along a single baseline to provide redundant information. This allows a simple matching algorithm to give very good results.

The principle of multi-baseline stereo is illustrated in Figure 1. In normal stereo, there are two images, a *reference* and a *match* image. For each pixel in the reference image, and for each possible disparity, the error is calculated between the reference and match images.

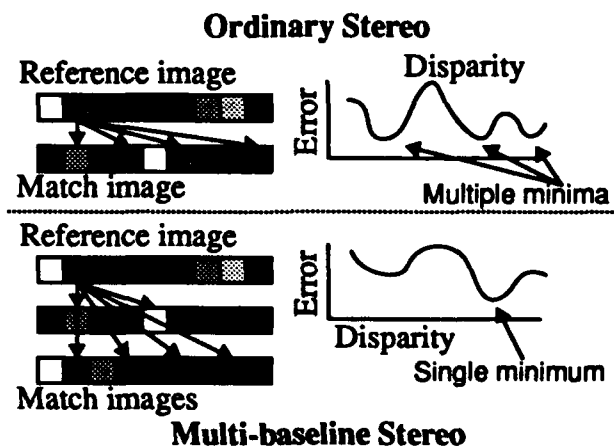


Figure 1. Ordinary and multi-baseline stereo.

Ordinary stereo can give multiple false matches, as illustrated. Multi-baseline stereo

reduces this problem by taking advantage of redundant information. As illustrated, we have arranged the cameras so that a disparity of d in the first match image corresponds to $2d$ in the second. It is unlikely that a pixel in the reference image will match pixels at corresponding disparities in both images unless the match is real.

The error is calculated by summing the error between the images over a small window of size $h \times w$. With a naive algorithm, this calculation over a image of size $r \times c$ will take $rchw$ add operations. By maintaining row and column sums, we can reduce this to $4rc$ operations plus some overhead. However this also reduces the available parallelism, because the row and column sums must be updated sequentially.

3. Parallel design issues

Adapt was used to implement the stereo vision algorithm. Adapt is a specialized language for image processing on parallel computers that has been extensively described elsewhere [3]. The Adapt compiler automatically parallelizes image processing programs, but requires the programmer to write them in terms of a series of *image to image* operations, each of which is described as a program to be applied at every pixel or every row of the images in parallel.

Multi-baseline stereo vision divides naturally into two steps, which are repeated for each disparity d :

1. *Difference*. Calculate the difference image, which is the pixel-by-pixel error between the reference image and the match images. This is done by forming the sum of squared differences between the reference pixel and the match images shifted within rows by appropriate multiples of d .
2. *Minimize*. At each pixel, sum the difference image over the $h \times w$ window and compare this with the best error so far. If less, replace the error value by the new value and the previous disparity by d .

The *difference* step can be implemented easily and efficiently in Adapt. However, if the *minimize* step is implemented in the obvious way processor utilization is poor. This is because the number of rows h in the window is generally large compared to the number of rows per processor; for example, the image size may be 240×256 and there are 64 processors, while the window size is 13×13 . When the window sum is formed, each processor will be repeating the work of other processors with overlapping rows. As a result, there will be little or no speedup as the number of processors is increased beyond the point at which the number of rows per processor equals the window height, or about 18 processors in this case.

The solution to this is to pre-calculate the partial sum of the difference image in the *difference* step. Each processor can simply add its four rows together, forming a *sum* image, which is distributed to other processors before the *minimize* step. The *minimize* step then forms its initial row sum by adding in appropriate rows of the *sum* image, together with whatever image rows are needed to make the window height exactly h rows.

4. Computation Issues

Table 1 summarizes the performance of the computation in the stereo program and compares the performance of the assembly-language routines used with C-generated code.

Language	Assembly	C
<i>Difference</i>	22.0 ms (257 MFLOPS)	36.0 ms (157 MFLOPS)
<i>Minimize</i>	17.3 ms (369 MFLOPS)	40.7 ms (156 MFLOPS)
Total	39.3 ms	76.7 ms

Table 1. Stereo Computation Times for three 240×256 images and 16 disparity levels, on 64 processors.

There are two reasons why the assembly code is significantly faster than the C code: (1) The compiler optimizes code primarily within a basic block. In assembly coding, there is no such restriction. (2) Bugs in the C-step imple-

mentation of the iWarp component, impact the performance of the compute-and-access instruction, which allows floating point add and multiply operations to proceed in parallel with memory access operations. Since the C compiler must generate code for the most general case, it cannot generate the most efficient code. However, within Adapt these bugs either cannot occur or can easily be avoided.

5. iWarp Communications Structure

Before discussing communications issues in the stereo program, we first discuss the communication capabilities of the iWarp computer; more detail is available elsewhere [1]. It is a two-dimensional torus, each of which is connected to its nearest neighbor by input and output ports, each of which can transfer data at 40 MB/s. Cell (0,0) of the processor array is connected to the SIB, a special iWarp processor that can communicate with I/O devices across a VME bus; all data processed by the iWarp array must pass through the SIB.

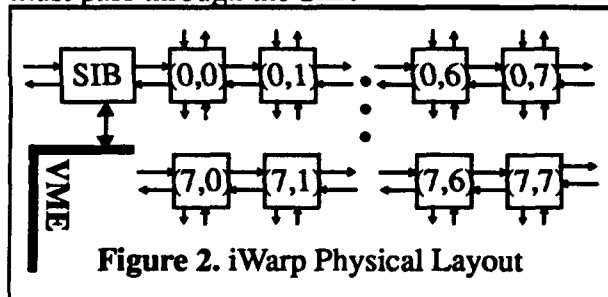


Figure 2. iWarp Physical Layout

iWarp supports *connections* which define communications routes among processors. Within connections *messages* can be used to allow more than two processors to communicate on a single connection. Within messages, data can be sent word by word (called *systolic*) or with a DMA-like operation called *spooling* which transfers the data in the background.

6. Communications Issues

The communications operations in Adapt and their theoretical maximum physical bandwidths are:

- **Broadcast:** duplicate data presently in the SIB on all processors. 40 MB/s.

- **Distribute:** distribute an image presently in the SIB to other processors, by row swaths. 80 MB/s.
- **Collect:** collect an image distributed on the processors by row swaths and transfer to the SIB. 80 MB/s
- **Create working copy:** obtain the grace bands needed for processing a swath of the image from adjacent processors. 160 MB/s.

These bandwidths are achieved (to within about 10%) in our implementation of Adapt on iWarp, even for relatively small amounts of data. To do this, we define primitive maximum bandwidth operations and then implement all communications operations in terms of them, ensuring maximum performance for communications. The primitive I/O for iWarp are:

1. **Receive.** Data is taken from the pathway and stored in memory.
2. **Send.** Data is taken from memory and sent to the pathway,
3. **Pass.** Data is passed from one pathway to another.
4. **Receive and pass.** Data is be passed from one pathway to another and simultaneously copied into memory.
5. **Receive and pass twice.** Data is be copied into memory and also sent to two other pathways.

6.1 Broadcast

Broadcast can be implemented in a variety of ways. The SIB is connected to the processor array at two points, so data can be sent either entirely from one connection or over both connections simultaneously. Once the data is inside array it can be transferred from cell to cell in a pipeline using *receive and pass* or in a tree fashion by also using *receive and pass twice*. We will consider these three techniques: using one SIB connection and pipelined transfer within the array (method (a)); using both SIB connections and pipelined transfer (method (b)); using one SIB connection and tree transfer within the

array (method (c)).

All of these methods achieve the same maximum bandwidth. The only difference between them is in latency. These differences in latency lead to a significant difference in performance, as illustrated in Figure 3.

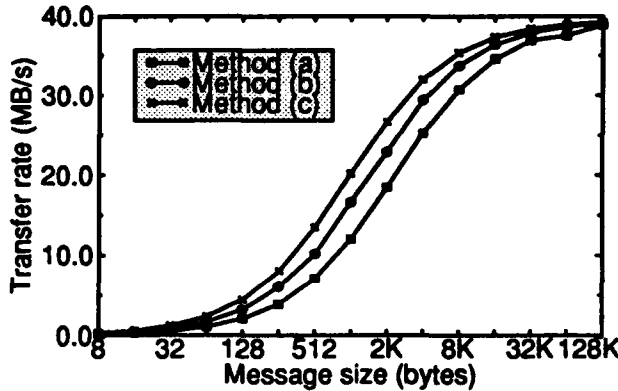


Figure 3. Performance of different implementations of *broadcast*.

Currently, Adapt uses method (a) for *broadcast*, because it was the easiest to implement.

6.2 Distribute

There are only two fundamentally different methods of implementing *distribute*. In the first method, data is sent systolically from the SIB over one pathway; each cell executes a *receive* followed by a *pass*. In the second method data is sent from the SIB over two pathways, one that goes in a forward direction through the processors and the other that goes in a reverse direction. Processors in the forward pathway act as before, while processors in the reverse pathway execute at *pass* followed by a *receive*.

Since the second method has a maximum physical bandwidth of 80 MB/s versus 40 MB/s for the first method, it is to be preferred if it does not impose too high startup overhead. We therefore examine the transfer rate for images of various sizes in Figure 4. From this graph we observe that even for small images of 8K bytes, the transfer rate exceeds the maximum that can be expected from using just one pathway. We therefore adopt the two pathway method.

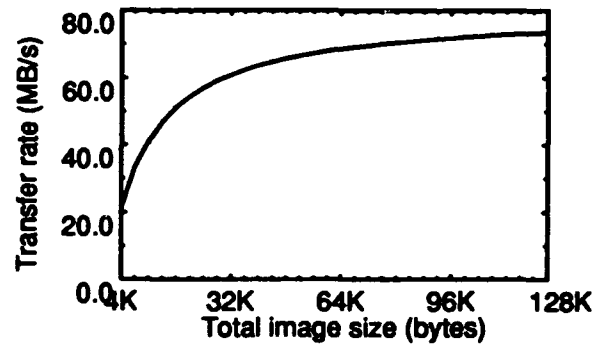


Figure 4. Performance of *distribute*.

6.3 Collect

Collect can be implemented using one or two pathways, just as with *distribute*. The performance of the two pathway method is shown in Figure 5. As with *distribute*, the graph rapidly exceeds the maximum bandwidth with a single pathway (at 12K bytes the bandwidth is 42 MB/s), so this method is chosen. However, the maximum bandwidth achievable in *collect* seems to be 60 MB/s, which is less than the theoretical maximum of 80 MB/s. This is due to a bug in the iWarp hardware that limits the maximum physical bandwidth while transferring into the SIB to 30 MB/s over any one physical pathway.

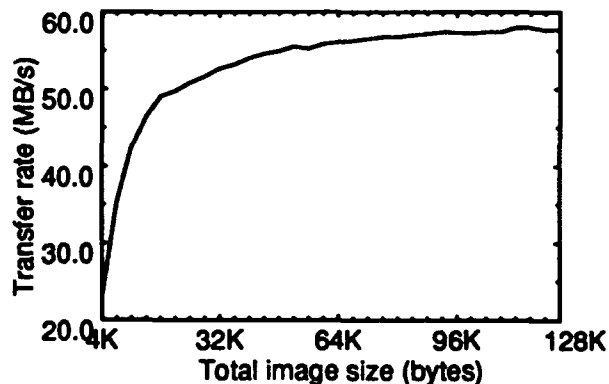


Figure 5. Performance of *collect*.

6.4 Create working copy

In *create working copy* each cell initially has several rows of data, and will obtain some rows of data above and below its rows from adjacent cells. The behavior is shown in Figure 6. From this figure, it is clear that *create working copy*

can be implemented as two *shift* operations, in which processor i sends its rows of data to the next or previous processor in the array, depending on whether the shift is in a forward or reverse direction. *Create working copy* consists of one shift in the forward direction and one shift in reverse.

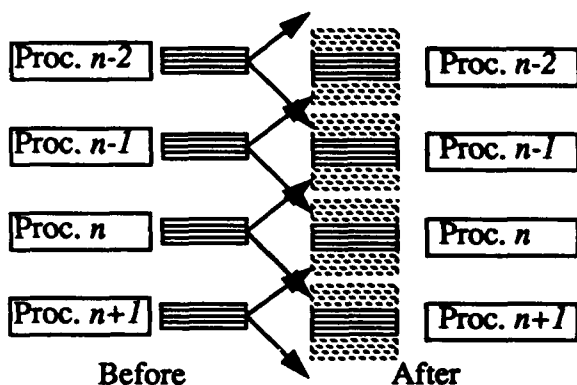


Figure 6. Create working copy.

Each *shift* operation can be implemented with *send* and *receive* operations in the obvious way. Using spooling, each processor can be sending and receiving data for both shifts simultaneously, giving a maximum bandwidth of 160 MB/s.

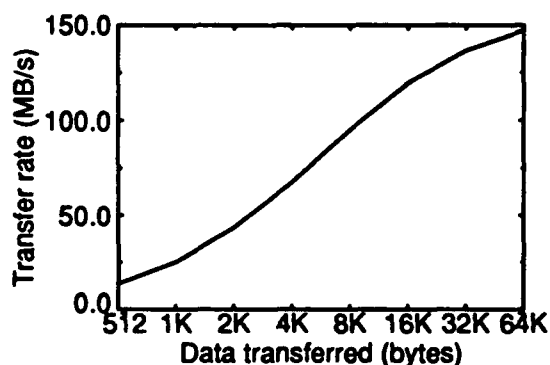


Figure 7. Performance of *create working copy*.

The performance of the I/O in *create working copy* is shown in Figure 7. The transfer rate seems to peak at 150 MB/s, instead of the expected 160 MB/s; some of this difference must be due to the same hardware limitation that led to the maximum 60 MB/s transfer rate to the SIB in *collect*.

6.5 Stereo I/O times

Table 2 summarizes the stereo I/O times. The times for *distribute* and *create working copy* were measured directly, and the times for *broadcast* and *collect* were estimated from the work earlier in this section (these times could not be estimated directly in a running system since they are overlapped with other computation; thus, the times reported here are overestimates in terms of the impact of these times on total execution time.)

<i>Broadcast</i>		1.92 ms
<i>Distribute</i>		3.00 ms
<i>Create working copy</i>	<i>difference image</i>	8.76 ms
	<i>sum image</i>	4.90 ms
<i>Collect</i>		4.16 ms
Total		22.7 ms

Table 2. Stereo I/O times for three 240×256 images and 16 disparity levels

7. Scaling with number of processors

Figure 8 gives the scaling of the stereo vision algorithm (for three 240×256 images with a 13×13 summation window) from 16 to 64 processors. The execution time does not decrease smoothly with increasing numbers of processors because the program is sensitive to the match between the number of rows in the image and the number of processors. This leads to a significant loss of efficiency with processor numbers of 40, 48, and 56.

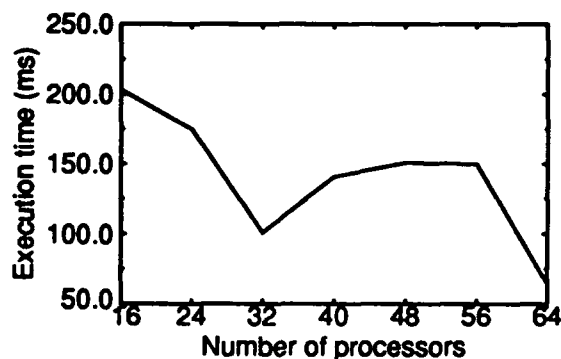


Figure 8. Scaling of performance with number of processors.

8. Conclusions and Future Work

The stereo vision system described here is the fastest system for obtaining depth data presently available — it is even faster than laser ranging systems, for a comparable number of depth measurements. Achieving this high performance was not simply a matter of reimplementing an existing algorithm on a parallel computer, but rather required rethinking of the algorithm in one step, and particular attention to I/O within the algorithm:

- The introduction of the *sum* image in the *difference* step made it possible to scale the algorithm beyond 18 processors.
- Thinking of the I/O in iWarp in terms of a collection of primitive operators that could be assembled to get the Adapt I/O functions was the key to efficiency. Previous implementations of the Adapt I/O functions made apparently reasonable choices (e.g., using a binary tree and message-passing to implement *broadcast*) but because there was no systematic approach to getting maximum efficiency performance was poor, and, more importantly, it was impossible to determine whether the implementation was as good as possible. In the work reported here we are reasonably certain that the Adapt I/O functions are as efficient as possible.
- Most parallel computers support only message passing, and have no systolic communication facilities. This limits the primitive operators to *receive* and *send*. Using just these operators, it is possible to build the Adapt I/O functions, but with great loss of efficiency, especially in *broadcast*. Perhaps by analysis of the primitive I/O functions the machine is capable of as it is being constructed, increased flexibility can be introduced, resulting in greater efficiency for common program communications functions.

Future work will concentrate on integrating the computation described here with the camera and display interfaces to demonstrate high-

speed stereo vision in the real world. The fastest VME-based framebuffer known to us is the Ironics IV-FCFG, which supports a 5.12 MB/s transfer rate, but requires transferring 4 bytes for every pixel (in RGOB format), which quadruples the data that must be transferred for display (the three input images can be transferred as one RGB image). Using this framebuffer, we have been able to achieve a cycle time for the stereo algorithm of 97 ms, including digitization, input, and output, for an image processing rate of 10 Hz. We are investigating other solutions that will allow us to achieve the 15 Hz processing time possible on iWarp; several solutions look promising.

Acknowledgments

The work on stereo vision was started by Hans Thomas and Bill Ross of the Vision and Autonomous Systems Center at Carnegie Mellon. The Adapt I/O library was originally written by Doug Smith of the Adapt project. Michael Mills of the iWarp project did some of the early work on the iWarp implementation of multi-baseline stereo. The support of several other members of the iWarp group, in particular Thomas Gross, Dave O'Hallaron, and Susan Hinrichs, is gratefully acknowledged. All graphs in this paper were prepared using ACE/gr, a public domain freeware tool written by Paul J. Turner of the Oregon Graduate Institute.

Bibliography

- [1] Borkar, S., et al. Supporting Systolic and Memory Communication in iWarp. *Proceedings of the 17th Annual International Symposium on Computer Architecture*. Seattle, Washington, May, 1990.
- [2] Okutomi, M., and Kanade, T. A Multiple Baseline Stereo. *Proceedings of the 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 63-69. IEEE Computer Society, Lahaina, Maui, Hawaii, June, 1991.
- [3] Webb, J. A. Steps Toward Architecture Independent Image Processing. *IEEE Computer*. 25(2):21-31, February, 1992.

Section XVI

**Reflectance
Polarization
Color**

Estimating Scene Properties from Color Histograms

Carol L. Novak

Siemens Corporate Research
755 College Road East
Princeton, NJ 08540

Steven A. Shafer

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

One of the key tools in applying physics-based models to machine vision has been the analysis of color histograms. We present here an algorithm for analyzing color histograms that yields estimates of surface roughness, phase angle between the camera and light source, and illumination intensity. In addition, an understanding of the effect of these parameters upon highlight appearance allows us to make better estimates of illumination color and object color.

We test our algorithm on simulated data and the results compare well with the known simulation parameters. We also test our method on real images and the results are reasonably close to the actual parameters estimated by other means. Our method for estimating scene properties is very fast, and requires only a single color image.

1. Introduction

Color histograms have long been used by the machine vision community in image understanding. Color is usually thought of as an important property of objects, and is often used for segmentation and classification. Unfortunately color is not uniform for all objects of a given class, nor even across a single object. Therefore color variation must be modeled in some way.

The earliest uses of color histograms modeled the histogram as a Gaussian cluster in color space [3]. In 1984 Shafer showed that for dielectric materials with highlights, the color histogram associated with a single object forms a plane [10]. This plane

is defined by two color vectors: a body reflection vector and a surface reflection vector. Every pixel's color is a linear combination of these two colors. In 1987 Klinker and Gershon independently observed that the color histogram forms a T-shape or dog-leg in color space [6], [2]. These histograms are composed of two linear clusters, one corresponding to pixels that exhibit mostly body reflection and one corresponding to pixels exhibiting surface reflection. This T-shape made it possible to identify characteristic body reflection and illumination colors.

However, there is more to be said about the shape of the color histogram. In previous work, we showed that color histograms have identifiable features that depend in a precise mathematical way upon such non-color scene properties as surface roughness and imaging geometry [9]. In this paper we show that three scene properties—the illumination intensity, the roughness of the surface, and the phase angle between camera and light source—may be recovered from three measurements of the histogram shape.

The functions that relate the scene properties to the histogram measurements are interdependent and highly non-linear. Since an exact solution is not feasible, we have developed a method that interpolates between data from a lookup table. Our work has also shown how the colors observed in a highlight depend upon more than just the scene colors, but also upon the surface roughness and the imaging geometry [9]. Thus, our estimates of these scene parameters allow us to improve initial estimates of the illumination color. This means we are better able to discount the effect of the illuminant to recover estimates of the object's reflectance.

We begin in section 2 with a brief explanation of the relationship between the color histogram fea-

tures and the various scene parameters. Section 3 presents an algorithm to compute estimates of these parameters from the histogram. Section 4 shows how our algorithm has been applied to real images.

2. Understanding Color Histograms

When we talk about the color histogram, we mean a distribution of colors in the three-dimensional RGB space. For a typical imaging system with 8 bits for each color band, there are 256^3 "bins" into which a pixel may fall. In this work, we only consider whether a bin is full or empty. We do not use a fourth dimension to record the number of pixels which have a particular RGB value. A fourth dimension would be difficult to visualize, but more significantly would also be dependent on such things as object size and shape.

In our work we use the "Dichromatic Reflection Model", where the light L reflected from a dielectric object is the sum of two components: a body component L_b and a surface component L_s [10]:

$$L(\lambda, \theta_i, \theta_r, \theta_p) = L_b(\lambda, \theta_i, \theta_r, \theta_p) + L_s(\lambda, \theta_i, \theta_r, \theta_p)$$

Each of the two components is a function of the wavelength of light (λ) and the angles of incidence (θ_i), reflection (θ_r), and phase angle (θ_p). The Dichromatic Reflection Model further states that each component in turn may be separated into a color term c that depends only on λ , and a magnitude term m that depends only upon θ_i , θ_r and θ_p :

$$L(\lambda, \theta_i, \theta_r, \theta_p) = c_b(\lambda) m_b(\theta_i, \theta_r, \theta_p) + c_s(\lambda) m_s(\theta_i, \theta_r, \theta_p)$$

Figure 1 shows a sketch of a typical color histogram for a dielectric surface illuminated by a single light source. As labeled, the histogram has two linear clusters of pixels: the body reflection cluster and the highlight cluster. The first of these clusters extends from the black corner of the cube (point a) to the point of maximum body reflection (point b). The other cluster starts somewhere along the body reflection cluster (point c) and extends to the highlight maximum (point d).

2.1. The Body Reflection Cluster

The linear cluster that we call the body reflection cluster corresponds to pixels that exhibit mostly

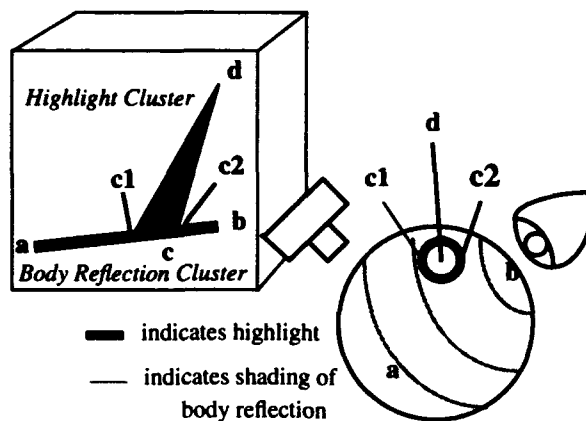


Figure 1: Histogram of an object

body reflection with very little surface reflection. If there is no ambient illumination in the scene, this cluster begins at the black point of the color cube (point a), corresponding to points on the surface with normals 90 degrees or more away from the direction of the illumination. The point at the other extreme of the body reflection cluster (point b), corresponds to the largest amount of body reflection seen anywhere on the object. Assuming that body reflection is Lambertian, the magnitude term will obey the relation

$$m_b = \gamma B_b \cos(\theta_i) \quad (1)$$

where θ_i is the angle of illumination incidence. The gain in converting photons measured by the imaging array into pixel values is represented by γ . The brightness of the body reflection is represented by the term B_b . This factors in both the reflectance of the object (albedo) and the intensity of the light.

If the object exhibits all possible surface normals, the body reflection cluster will be full length and densely filled. If the object is composed of a small number of flat surfaces, there will be gaps in the body reflection cluster. For this paper we will assume that objects we are looking at have a broad, continuous distribution of surface normals.

A vector fitted to the body reflection cluster (from point a to b) will point in the direction of the body reflection color, which is the product of the object color and the illumination color. If the illumination color has been determined from analysis of the highlight, the object color may be calculated by dividing out the illumination color, as shown in some color constancy methods [1], [4], [5], [11].

If we assume that there is some point on the object

which has a surface normal pointing directly at the light source (and which is visible to the camera), then at that point $\cos(\theta_i) = 1$. This means that the length of the fitted vector (the magnitude $|\overline{ab}|$) corresponds to the gain times the object's apparent brightness B_s . If the intensity of the illumination is also recovered from highlight analysis, then the object albedo can be separated from the object's apparent brightness (assuming that the gain of the camera had been calibrated). This makes it possible to tell a bright light shining on a dark surface from a dim light shining on a bright surface.

2.2. The Highlight Cluster

The cluster of pixels we call the highlight cluster corresponds to pixels that show a non-negligible amount of surface reflection. These pixels come from the areas of the image that form the highlight. In the histogram, the highlight cluster starts where it intersects with the body reflection cluster (point *c* in Figure 1) and extends upwards from there to the brightest point of the highlight (point *d*).

In this presentation we use the Torrance-Sparrow model of scattering [12]. This models a surface as a collection of tiny facets, with σ describing the standard deviation of facet angles with respect to the global surface normal. Smooth surfaces will have a small standard deviation while rougher surfaces will have a larger standard deviation. The equation for scattering gives the amount of surface reflection as

$$m_s = \gamma B_s \frac{FG(\theta_i, \theta_r, \theta_p) \alpha}{\sigma \cos(\theta_r)} \exp\left(-\frac{\theta_s^2}{2\sigma^2}\right) \quad (2)$$

where θ_s is the off-specular angle and θ_r is the angle of reflectance. B_s is the intensity of the illumination, γ is the camera gain, and α is a constant that includes the facet size (a variable in the original Torrance-Sparrow model). G is the attenuation factor and is a complicated function of imaging geometry (see [12] for details).

F is the Fresnel coefficient that describes the percentage of the light that is reflected at the interface; it is a function of geometry, wavelength, polarization state, and index of refraction of the material in question. However it is very weakly dependent on incident angle and wavelength (over the visible range), so we will follow the Neutral Interface Reflection model and assume that it is constant for

a given material [7]. Furthermore, for a wide range of plastics and paints, the indices of refraction are very nearly identical. For this paper we will assume that materials have an index of refraction of 1.5, corresponding to 4.0% Fresnel reflectance.

2.2.1. Length of Highlight Cluster

When looking at highlights on a variety of surfaces, we quickly observe that highlights are brighter and sharper on some surfaces, while they are dimmer and more diffused on other surfaces. Very shiny surfaces exhibit only a tiny amount of scattering of the surface reflection, whereas very matte surfaces have a great deal of scattering.

We see from equation (2) that the sharpness of the peak is determined by the standard deviation σ , and that the height of the peak is inversely proportional to σ . Intuitively this makes sense, since surface reflection scattered over a smaller area will be more "concentrated." A smooth object will have a small standard deviation of facet slopes (σ) resulting in a long highlight cluster. A rough object will have a large σ , and so will exhibit a shorter cluster.

Equation (2) indicates that the intensity of the light source B_s also affects the magnitude of the surface reflection, and thus the length of the highlight cluster. It is obvious from equation (2) that the length is directly proportional to this brightness.

Equation (2) also predicts that imaging geometry will have an effect upon highlight magnitude, as indicated by the $\cos(\theta_r)$ term in the denominator and the attenuation term G in the numerator. The effect on the length of the highlight cluster is small but noticeable, causing the length to increase as the phase angle is increased.

2.2.2. Width of Highlight Cluster

Another difference between histograms for smooth and rough surfaces is the width of the highlight cluster where it meets the body reflection cluster (the distance from point *c*₁ to point *c*₂ in Figure 1). The highlight cluster will be wider for rougher surfaces, and narrower for smoother surfaces. This is because rougher objects will scatter surface reflection more widely, over a larger number of reflectance angles.

In the color histogram, any amount of surface reflection results in pixels that are displaced from

the body cluster in the direction of the illumination color. If we take any highlight pixel and project along the surface color vector onto the body reflection vector, we can tell how much body reflection is present in that pixel. If we consider all the pixels in the highlight area of the image and look at how much body reflection is in each of them, we obtain a range of body reflection magnitudes. The rougher the surface, the larger the range of body reflection magnitudes, since the highlight is spread over a larger number of surface normals. This property is independent of the object's size and shape.

The brightness of the illumination, B_i , will also have an effect on the width of the highlight cluster. As the light intensity is increased, points on the surface that had amounts of surface reflection too small to be noticed may become bright enough to be included with the highlight pixels. Clearly the width will grow as the light intensity grows. However the growth is very slow, much slower than the linear growth of the length with increasing illumination intensity. This means that the ratio of the cluster length to the width grows as the illumination is increased, making it possible to distinguish a bright source illuminating a rough object from a dim source illuminating a shiny one.

Although the width of the highlight cluster does not depend upon the object's size and shape, it does depend upon the phase angle. To see why, consider a highlight that spreads 15 degrees in every direction from its maximum. If the camera and light source are separated by 30 degrees, the perfect specular angle will be at 15 degrees with respect to the illumination direction. The highlight will spread over points with surface normals ranging from 0 degrees to 30 degrees. (For this explanation, we will ignore the $1/\cos(\theta_r)$ term in equation (2).) The amount of body reflection at these points will vary from $\cos(0) = 1.0$ to $\cos(30) = 0.87$, a width of 0.13. On the other hand, if the camera and light source are separated by 90 degrees, the perfect specular angle will be at 45 degrees, with the highlight spreading from 30 degrees to 60 degrees. Now the amount of body reflection will vary from $\cos(30) = 0.87$ to $\cos(60) = 0.50$, giving a much larger width of 0.37.

2.2.3. Intersection of Clusters

When we introduced the diagram in Figure 1, we

described the highlight cluster as beginning "somewhere" along the body reflection cluster. Now we will show how to pinpoint the location. The distance along the body reflection cluster where the two clusters intersect (the length of \overline{ac} divided by the length of \overline{ab} in Figure 1) shows the amount of body reflectance at those points on the surface that are in the highlight. Assuming that body reflection is Lambertian, the amount of body reflection is proportional to the cosine of the incidence angle. If the two clusters intersect at the maximum point on the body reflection cluster, it means the highlight occurs at those points that have the maximum amount of body reflection, where surface normals point directly at the light source. If the two clusters meet halfway along the body reflection cluster, the highlight must occur at points with surface normals pointing $\arccos(1/2)$ or 60 degrees away from the illumination direction.

If the body reflection is Lambertian, it does not depend in any way upon the angle from which it is viewed. Thus the body reflection does not tell us anything about the camera direction. However, the surface reflection is dependent upon both the illumination and camera directions. If we ignore for a moment the $1/\cos(\theta_r)$ term in equation (2), we see that the maximum amount of surface reflection will occur at those points on the surface where the angle of incidence equals the angle of reflection. Thus if the highlight occurs at a point where the surface normal faces 10 degrees away from the light source direction, the light source and camera must be 20 degrees apart with respect to that point on the surface.

The $1/\cos(\theta_r)$ term in equation (2) means that the maximum amount of surface reflection will not always occur precisely at the perfect specular angle. This is particularly true of rougher surfaces where the highlight is spread over a wide range of reflectance angles, causing significant variations in $1/\cos(\theta_r)$. This causes "off-specular peaks" described in [13]. The result is that the intersection is slightly dependent upon the surface roughness.

2.2.4. Direction of Highlight Cluster

The highlight cluster is usually long and narrow in shape and a vector can be fitted to it (from point c to d in Figure 1). Klinker argued that this vector will usually correspond closely to the surface

reflection color [5]. This is true for smooth objects where the highlight has a small area, and for imaging geometries where the body reflection changes slowly over that area. In this case, the amount of body reflection at the base of the highlight cluster and the amount at the tip varies by a small amount.

On the other hand, if the object is rough and the highlight occurs on a part of the object where the cosine of the incidence angle changes more rapidly, then the amount of body reflection at the base of the highlight cluster may vary significantly from the amount at the tip. This has the effect of skewing the highlight cluster away from the direction of the illumination color. The estimate of the illumination color made from fitting a vector to this cluster will be somewhat inaccurate.

An inaccurate estimate of illumination color will, in turn, bias estimates of the object color. The vector fitted to the highlight cluster is a reasonable first estimate of illumination color, but we now know that it may be skewed. If we know the surface roughness and imaging geometry, we can calculate the amount of skewing and compensate for it.

In sections 2.2.1 and 2.2.2 we showed that the roughness of the object affects the length and width of the highlight cluster, but that there is some dependence on imaging geometry. Section 2.2.3 showed that the imaging geometry determines the intersection of the two clusters, but that there is some dependence upon roughness. Furthermore, the intensity of the illumination affects the length and width of the highlight cluster as well, although in different ways. The degree of dependence of each histogram measurement upon the scene parameters is characterized in Table 1.

	Roughness	Phase Angle	Illumination Intensity
Length	strong	weak	strong
Width	strong	strong	weak
Intersection	weak	strong	none

Table 1: Dependence of histogram features upon scene parameters

In this section we have shown that the direction of the highlight cluster may be skewed away from the direction of the illumination color, depending upon the roughness and the phase angle. The highlight cluster length, width and intersection are defined

with respect to the highlight cluster direction, so measurements of these features depend upon the estimate of the illumination color. If that direction is incorrect, the other histogram measurements will be affected. Obviously these factors are interdependent. Therefore we solve for the roughness, phase angle, and illumination intensity simultaneously.

3. Analyzing Color Histograms

The previous section described the relationship between scene parameters and histogram features. Understanding the relationship is the first step in analyzing color histograms. The next step is figuring out how to actually exploit the histogram to recover quantitative measures of scene properties.

The known image parameters which can be measured from the color histogram are the body reflection cluster's length and direction; the highlight cluster's length, width, and direction; and the intersection point of the two clusters. This gives four scalar values and two vector quantities. They will be referred to by the following variables:

- l - length of highlight cluster
- w - width of highlight cluster
- i - intersection of two clusters
- b - length of body reflection cluster
- \vec{d}_s - direction of highlight cluster
- \vec{d}_b - direction of body reflection cluster

The unknown scene parameters which we would like to recover from the histogram can also be divided into scalar values and vector quantities. These variables are:

- σ - optical roughness
- θ_p - phase angle
- B_s - illumination intensity
- B_o - object albedo
- \hat{c}_s - chromaticity of light source
- \hat{c}_o - object chromaticity under "white" light

For convenience, we have separated the illumination and object reflectance into intensity components and chromatic components. As we mentioned in section 2.1 the object's chromaticity \hat{c}_o may be recovered in a straightforward way from the direction of the body reflection cluster \vec{d}_b and the color of the light source \hat{c}_s . The red component of the body reflection vector is divided by the red

component of the light source color, the green component divided by the green component, and the blue component divided by the blue component. The result is normalized to length 1.

The albedo of the object B_o is also recovered in a straightforward manner from the length of the body reflection cluster and the illumination intensity. The length b is equal to the maximum amount of body reflection, where $\cos(\theta_r) = 1$. The object albedo B_o is simply the cluster length b divided by the illumination intensity B_s and by the gain γ .

For the remaining unknowns the situation is not so simple. From section 2 we know that the length l is related to surface roughness and illumination intensity, but is also dependent upon imaging geometry. The remaining knowns (l, w, i, \hat{d}_s) and unknowns ($\sigma, \theta_p, B_s, \hat{c}_s$) will be examined in detail in the next few sections.

3.1. Exact Solutions

Equations (1) and (2) describe the amounts of body and surface reflection, m_b and m_s , as a function of imaging geometry and light intensity. Equation (2) also shows how the amount of surface reflection varies with the roughness σ . Unfortunately it is not possible to directly solve for these scene parameters from the histogram measurements. For example, the length of the highlight cluster indicates the maximum amount of surface reflection seen anywhere. For given values of σ, θ_p , and B_s , the length l may be calculated

$$l = \text{MAX} \left[\gamma B_s \frac{FG(\theta_r, \theta_p, \theta_s) \alpha}{\sigma \cos(\theta_r)} \exp \left(-\frac{\theta_s^2}{2\sigma^2} \right) \right] \quad (3)$$

over all values of θ_r, θ_p , and θ_s . However, since G contains several trigonometric functions and the roughness term σ occurs both inside and outside the exponential, there is no analytic solution for σ from the length l .

Although equation (3) has no analytic solution, it might be possible to solve it iteratively, through some sort of search (for example by gradient descent). However that assumes the length l can be accurately measured from the histogram. In section 2.2.1, the length is measured from the tip of the highlight cluster to its base *along the direction of the highlight color*. Unfortunately the highlight color is not typically known in advance. As

described in section 2.2.4, the highlight color may be estimated by fitting a vector to the pixels that form the highlight cluster. However that cluster may be skewed.

This is shown in Figure 2, which is the histogram of a simulated rough object illuminated with white light. The dotted "measured" line shows the *direction* calculated for the best fit vector to the highlight cluster. (In this case the best fit line to the cluster will not pass through the brightest point in the highlight cluster.) The *position* of the dotted line in Figure 2 shows the projection of the brightest highlight pixel onto the body reflection vector *along the best fit vector*, since the length is calculated from the brightest pixel. The "ideal" line indicates the direction of the actual illumination color. The skewing causes the measured length to be longer than it would have been if the correct illumination color had been known.

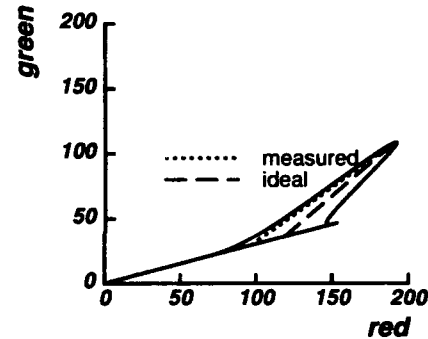


Figure 2: Skewed length measurement

This contrasts the "ideal" length that we would like to obtain from the histogram, with the "measured" length that can actually be recovered without *a priori* knowledge of the illumination color. When the highlight is skewed, the values measured from the histogram will be different from the ideal values that would be calculated from analyzing equation (2). In the absence of *a priori* information, we can only measure what is available in the histogram. How do we derive the ideal values from the ones that are measured? Once that is done how do we recover the image parameters in which we are interested?

3.2. Approximate Solution

Our approach is to recover scene parameters by an approximate method, *directly* from the initial histogram measurements. Therefore, we do not need to

recover “ideal” histogram values from the “measured” ones. The ideal values of the histogram are a useful abstraction since their relationship to scene parameters is easy to explain. However they cannot be obtained from the histogram without knowledge of the illumination color.

Figure 3 shows the variation in the measured length as roughness and phase angle are changed. Each value of length describes a contour within the space of roughness and phase angles. Given a length measurement from a histogram, the associated scene parameters must lie somewhere on that contour. When illumination intensity is considered along with roughness and phase angle, these three scene parameters form a three dimensional parameter space. A length measurement would then describe a two dimensional surface within that space, showing the possible roughness, phase angle, and light intensity values that could give rise to a histogram with that length highlight cluster.

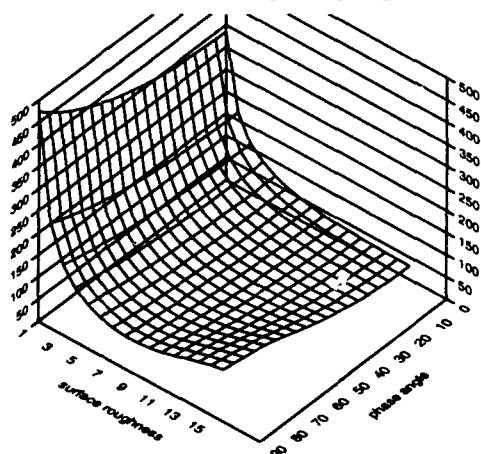


Figure 3: Variation in cluster length with changes in roughness and phase angle

The intersection and width measurements will also describe surfaces within the parameter space. The hope is that the surfaces for each of the histogram measurements will intersect at a single point in the parameter space, making it possible to recover unique values for surface roughness, phase angle, and illumination intensity. So obvious questions are: how can we generate these contours of equal-length, equal-width, and equal-intersection; and do these contours intersect to give unique solutions?

Section 3.1 pointed out that there is no analytic solution to generate the contours. However, Figure 3 shows how the highlight cluster length varies with roughness and phase angle at discrete

points. These values come from simulating an object with those parameters and then measuring the length, width and intersection of the highlight cluster in the resulting color histogram. By simulating a large range of roughness values, phase angles, and illumination intensities, we create lookup tables of length, width, and intersection measurements. We then search through the lookup table to find the scene parameters that correspond to a given set of histogram measurements.

The other question is whether a unique solution exists for a given triple (l, w, i) . If some triple has more than one solution, that means that different combinations of scene parameters can give rise to identical histogram measurements. It also means that a search through the contours in parameter space cannot be guaranteed to converge. We have explored the distribution of possible (l, w, i) triples, and found that each set of measurements is associated with at most one set of scene parameters. (Of course, many points within the measurement space will not correspond to any set of scene parameters.) The only remaining problem is to determine *which* set of scene parameters is associated with a given measurement triple.

3.3. Generating Lookup Tables

The range of roughness values, phase angles, and illumination intensities used to create the lookup table is shown in Table 2. The roughness value is the standard deviation of facet angles with respect to the global surface normal. The phase angle is the angle between the camera and light source with respect to the object. The light intensity is a percentage of a hypothetical light's maximum output.

	Min	Max	Increment	Total Used
Roughness	1°	15°	2°	8
Phase Angle	0°	90°	10°	10
Intensity	50%	100%	10%	6
Overall	480			

Table 2: Range of parameters

For each set of roughness, phase angle, and light intensity values, a simulated object is generated. The histogram associated with the object is automatically separated into body reflection and highlight clusters. The technique is similar to that described in [5]. Vectors are fitted to each of the

clusters.

Once the direction of the highlight cluster has been measured, the vector \vec{d}_s is used to project all highlight pixels onto the body reflection vector \vec{d}_b . These projections determine the relative contributions of the vectors in each pixel. Each color pixel \vec{p} in the histogram can then be defined as

$$\vec{p} = m_s \vec{d}_s + m_b \vec{d}_b$$

This is essentially the dichromatic equation, although the highlight cluster direction \vec{d}_s may differ from the actual surface color. The histogram measurements are then defined simply as

$$l = \text{MAX} (m_s) \text{ over all } \vec{p} \quad (4)$$

$$b = \text{MAX} (m_b) \text{ over all } \vec{p} \quad (5)$$

$$i = m_b / b \text{ for that } \vec{p} \text{ with maximum } m_s \quad (6)$$

$$w = [\text{MAX} (m_b) - \text{MIN} (m_b)] / b \text{ over all } \vec{p} \text{ for which } m_s > T \quad (7)$$

The threshold T is set according to the noise level of the camera.

3.4. Calculating Roughness, Phase Angle, and Illumination Intensity

Once the length, width and intersection have been measured, the problem is to determine which scene parameters will give rise to that shape histogram. Our work uses a polynomial approximation to the lookup table data. We assume that the roughness can be approximated as a polynomial function of the length, width, and intersection measurements of the histogram.

$$\begin{aligned} \sigma &\approx f_n(l, w, i) \\ &\approx A + Bl + Cw + Di + \\ &\quad El^2 + Fw^2 + Gi^2 + Hlw + Ili + Jiw + \dots \end{aligned} \quad (8)$$

The lookup table provides the means for calculating the coefficients of the polynomial. It provides almost 500 sets of histogram measurements and the associated roughness values. Least squares estimation is used to calculate the best fit n th degree polynomial to the data. A fourth degree polynomial is used in our experiments.

Similarly, the phase angle and illumination intensity are also approximated as polynomial functions of the histogram length, width, and intersection.

$$\theta_p \approx g_n(l, w, i) \quad (9)$$

$$B_s \approx h_n(l, w, i) \quad (10)$$

Generating the lookup table is obviously very time consuming (about 8 hours on a SPARC II) since it involves calculating almost 500 graphics simulations. However, the table generation and coefficient calculation only need to be done once and can be done ahead of time. At run-time our system takes a histogram from an image with unknown parameters and automatically separates it into two clusters and measures their dimensions. The polynomial equations are then applied to quickly estimate the roughness, phase angle, and illumination intensity. The run-time portion is very quick, taking less than 3 seconds on a histogram containing about 3600 pixels. If the histogram has already been split into clusters in the process of segmenting the image [6], the time to calculate the scene parameters is less than 1 second.

To test the polynomial approximations, one hundred test images were simulated and then analyzed by our method. The surface roughness, phase angle, and illumination intensity values used in the test images were chosen by a pseudo-random number generator. The test values were constrained to lie within the ranges used in the lookup table.

The calculated values of σ , θ_p , and B_s were compared with the original values used to generate the image. In almost all cases the calculated values were close to the original ones. However, for 2% of the cases, the values were very obviously wrong. For example, a negative value of roughness or illumination intensity is clearly unreasonable. Fortunately, bad values can be detected automatically, by checking to see if recovered values are within the allowable range. Recovered values that fall outside that range indicate that a different method should be used to recover the scene parameters. We have found that the problem disappears if a third degree polynomial is used instead, although the overall error on all cases is slightly higher.

Table 3 shows the results for the remaining 98 cases where the fourth degree polynomial produced reasonable estimates. It shows the average error in recovering the parameter, and also reiterates the step sizes used in the table. The errors are lower than the table resolution, showing the interpolation method is fairly effective.

The results for calculating roughness and phase

Parameter	Average Error	Table Resolution
Roughness	1.20°	2°
Phase Angle	4.40°	10°
Intensity	8.18%	10%
Cases Considered	98/100	

Table 3: Results on simulated data

angle are very good. They show that these non-color parameters may be calculated with reasonably high accuracy just by considering the shape of the color histogram. The error in calculating illumination intensity is a bit higher, although it still provides a useful estimate.

3.5. Calculating Illumination Chromaticity

As we pointed out in section 2.2.4, the highlight cluster may be skewed from the direction of the illumination color. The skew is particularly pronounced for large phase angles and rough surfaces. These two factors determine how much the body reflection changes over the area of the highlight.

Therefore, if we know or can calculate the surface roughness and the imaging geometry, we can in turn calculate the amount of highlight skewing. Once the skew is known, its effect can be subtracted from the direction of the highlight cluster to give the true color of the illumination.

Section 3.4 showed how to estimate the roughness and phase angle from the color histogram. These estimates are now used to estimate the skew. Again, a lookup table approach is used. When the simulations are performed to fill the lookup tables with measurements of length, width, and intersection, the skewing of the highlight is also calculated and stored in the table. Then a polynomial function is used to calculate the skew angle as a function of roughness, phase angle, and illumination intensity:

$$\text{Skew} \approx \Lambda_n(\sigma, \theta_p, B_s) \quad (11)$$

A third degree polynomial is used in our experiments.

Once the skew has been calculated, the illumination color \hat{c}_s may be calculated from the measured direction \hat{d}_s and the calculated skew angle. Obviously, if the polynomial functions described in section 3.4 produce bogus estimates of the roughness, phase angle, or illumination intensity, there is little point in plugging them into the equation for

calculating *Skew*. In those 2% of the 100 test cases, the program did not attempt to calculate the illumination color. For the remaining 98 test cases, the skew angle was used to calculate the illumination color. The results are shown in Table 4. The error in estimating skew is the difference between the correct skew angle and the skew angle calculated by our method. The table shows the average error over the 98 cases considered. It also shows the minimum, maximum and average of the actual skew values. For 69 of the test images, the scene parameters were such that the highlight was skewed by more than 1°.

Average error	1.73°
Average skew	8.63°
Minimum value	0.01°
Maximum value	27.5°
Number of skews > 1°	69
Cases considered	98/100

Table 4: Results in calculating skew

We will describe our algorithm's performance on one of the test simulations. The test image is a red cylinder under white light. The simulation parameters are given in the first column of Table 5. The histogram associated with this image is shown in Figure 4. The program automatically divided the histogram into body reflection and highlight clusters. The measurements made of the histogram are shown in the second column of Table 5.

Simulated Image	Histogram Measurements	Recovered Parameters
$c_s =$ [0.58, 0.58, 0.58]	$d_s =$ [0.81, 0.45, 0.37]	$c_s =$ [0.59, 0.57, 0.57]
$\sigma = 12.06^\circ$	$l = 74.1$	$\sigma = 11.99^\circ$
$\theta_p = 63.30^\circ$	$w = 0.47$	$\theta_p = 67.05^\circ$
$B_s = 90\%$	$i = 0.51$	$B_s = 89\%$

Table 5: Example results

The direction fitted to the highlight cluster is significantly skewed away from the direction of the actual illumination color. It represents a much redder color than the white illumination color, and so would be a poor estimate of the illumination color. It would also yield an inaccurate estimate of the object color when the illumination color is divided out of the body reflection color.

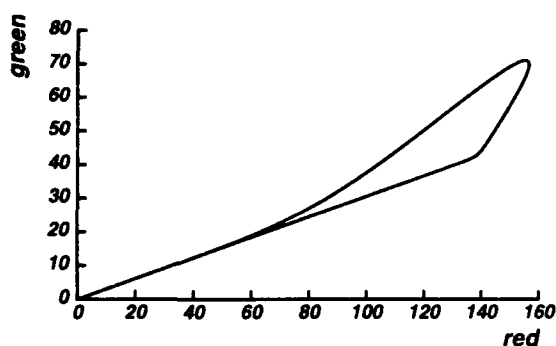


Figure 4: Histogram of simulated image

Applying polynomial equations (8), (9) and (10) to the length, width and intersection measurements, the program estimated the scene parameters shown in the third column Table 5. We then applied equation (11) to these estimates of σ , θ_p , B_s , and estimated the skew between the highlight cluster direction and the actual illumination color to be 18.75° . Applying this skew to the cluster direction \hat{d}_s , produced an estimate of the light source chromaticity that is very close to the original white color. The complete algorithm is diagrammed in Figure 5.

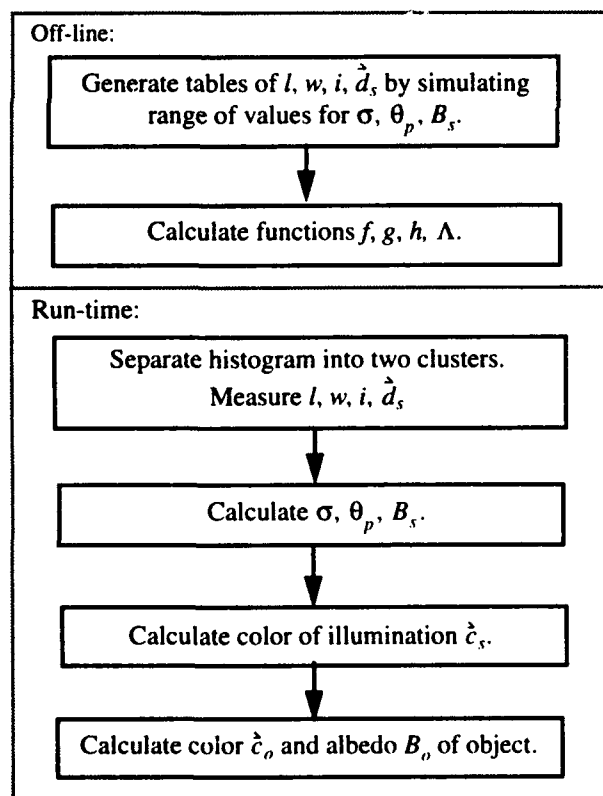


Figure 5: Algorithm for calculating scene parameters

4. Applying the Algorithm to Real Images

Real images present many challenges for vision researchers. These include camera noise, clipping, chromatic aberration, blooming, color imbalance, etc. We have modified our algorithm in systematic ways to adapt to these conditions. These modifications are described in [8]. We will describe here some experiments we have performed to test our algorithm on real images.

4.1. Estimating Phase Angle

An experiment was set up in the Calibrated Imaging Laboratory (CIL) at Carnegie Mellon University to test our algorithm's ability to estimate phase angle from real images. A series of images was taken with the camera and light source separated by an increasing phase angle. The angle was estimated with a large protractor and strings to indicate the direction of the camera and light source. The angles measured by this method were estimated to be accurate to within 5 degrees.

The first image in the sequence was taken when the camera and light source were approximately 10 degrees apart. The phase angle was then increased by 10 degrees between each picture. The last image in the sequence was taken when the phase angle between the camera and light source was 90 degrees.

The program automatically split the color histogram of the object into two clusters, fit vectors to those clusters, and calculated the values of length, width, and intersection. This was repeated for each image in the sequence. The polynomial approximation described in section 3.4 was used to calculate the phase angle from the length, width and intersection measurements. The results are shown in Figure 6. The dotted line shows the correct answer, using the phase angle measured by the protractor as ground truth. The average error in estimating angle is 9.96° .

Overall the method developed for estimating phase angle from analyzing color histograms works fairly well, especially considering that the ground truth measurement of the phase angle is fairly crude. Also the lookup tables were calculated without calibrating the simulated images to the conditions in the CIL. In particular, the noise of the camera was not measured precisely and the light source used in

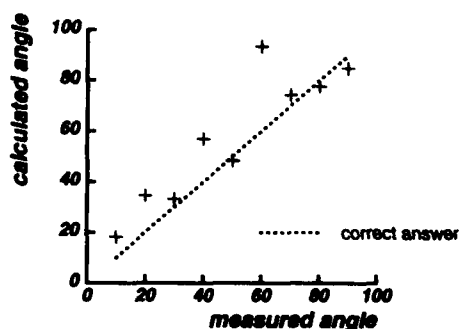


Figure 6: Results for calculating phase angle from real images

the experiments was not a point source as was used in the simulations.

4.2. Estimating Illumination Intensity

A second experiment was performed in the CIL to test the performance of the algorithm at estimating illumination intensity. The light was plugged into a variable voltage supply with a manually operated dial. A sequence of images was taken under increasing levels of illumination, while the imaging geometry and target object were kept constant. Altogether six images were taken. The illumination level was measured with a spot meter aimed at a white card. The spot measurements were estimated to be repeatable to within 5%.

Again, the program analyzed the histograms to produce measurements of length, width and intersection for each image. The polynomial equation to calculate illumination intensity was then applied to these measurements. The results are shown in Figure 7. The horizontal axis shows the values produced by the spot meter, while the vertical axis shows the intensity estimated by the histogram analysis. The gain of the camera has not been calibrated, so the program gives a relative estimate of intensity. The dotted line shows the best linear fit to the data. If the slope of that line is considered to be the gain of the camera, then the average error in estimating illumination intensity is 5.07%.

4.3. Estimating Roughness

A third experiment was performed to show how the system estimates surface roughness from color histograms. Figure 8 shows a composite of five images of different objects. The upper left shows a green plastic toy in the shape of an alligator; the upper right contains an orange plastic pumpkin for

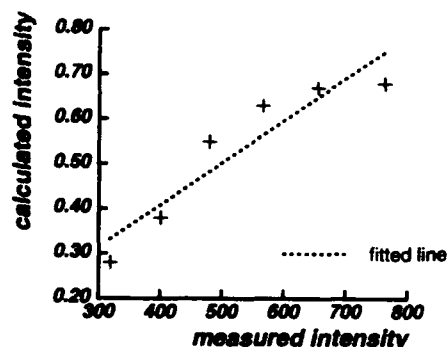


Figure 7: Results for calculating illumination intensity from real images

trick-or-treating; in the lower left is a terra-cotta ball; in the lower right a red plastic beach ball. In the center is a red plastic pail.

Table 6 shows the roughness calculated by the system for each of these objects. The objects are listed in order of decreasing roughness, as estimated by human observation. The calculated roughness number is the standard deviation of facet angles.

Object	Calculated Roughness
Alligator	10.07°
Pumpkin	8.93°
Terra-cotta ball	3.61°
Red ball	0.40°
Red pail	0.10°

Table 6: Results for estimating roughness

There is no error measure for these results, since there is no ground truth data for the actual roughness values. Nevertheless, the roughness ranking from the program agrees with that produced by a human observer. The pumpkin presents a particu-

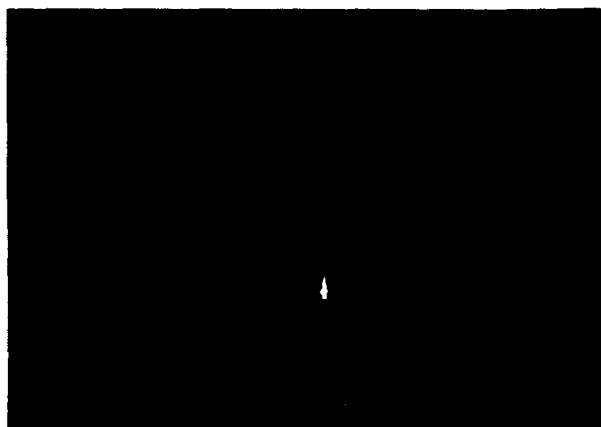


Figure 8: Five objects with different roughness values

larly interesting case, since it shows roughness at more than one scale. At the large scale where roughness is judged by touch, it is clearly the roughest object. This type of roughness is large enough to be considered "texture". At the smaller scale of optical roughness, it is considered to be more shiny than the alligator.

5. Conclusions

The color histogram of an image is a rich source of information, but it has not been fully exploited in the past. We have shown that the color histogram of a dielectric object may be characterized by a small number of measurements, which relate directly to many scene properties. We have shown how these histogram measurements may be used to recover estimates of surface roughness, imaging geometry, illumination intensity, and illumination color. These estimates may in turn be used to calculate object color and albedo.

The resulting algorithm is applied to real images, and produces reasonable estimates of phase angle, illumination intensity, and surface roughness. The method is independent of the shape of the object, and works on shapes ranging from a pumpkin to an alligator. The model used to develop the lookup tables is fairly general, and was not calibrated to match the actual imaging conditions such as light source extent, camera noise characteristics, etc. This kind of analysis may be applied to such varied tasks as surface inspection and object recognition.

6. Acknowledgments

The authors would like to thank Reg Willson, Jim Moody and Bill Ross who developed much of the hardware and software that makes picture-taking possible in the Calibrated Imaging Laboratory.

This research was sponsored by the Avionics Lab, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597; and by the Air Force Office of Scientific Research under Contract F49620-86-C-0127.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Govern-

ment.

Bibliography

- [1] D'Zmura, M. and Lennie, P. "Mechanisms of color constancy." *J. Opt. Soc. Am.*, 3(10) 1986, 1622-1672.
- [2] Gershon, R. *The Use of Color in Computational Vision*. Ph.D. thesis, University of Toronto, 1987.
- [3] Haralick, R. M. and Kelly, G. L. "Pattern recognition with measurement space and spatial clustering for multiple images." *Proceedings IEEE*, 57, 1969, 654-665.
- [4] Healey, G. and Binford, T. O. "The role and use of color in a general vision system." *Proc. of the ARPA IU Workshop*, 1987.
- [5] Klinker, G. J. *A Physical Approach to Color Image Understanding*, Ph.D. thesis, Carnegie Mellon University, 1988.
- [6] Klinker, G. J. and Shafer, S. A. and Kanade, T. "Using a color reflection model to separate highlights from object color." *Intl. Conf. on Computer Vision*, IEEE, 1987, 145-150.
- [7] Lee, H. C. "Estimating the illuminant color from the shading of a smooth surface." AI Memo 1068 MIT AI Lab. 1988.
- [8] Novak, C. L. *Estimating Scene Properties by Analyzing Color Histograms with Physics-Based Models*, Ph.D. thesis, Carnegie Mellon University, 1992.
- [9] Novak, C. L. and Shafer, S. A. "Anatomy of a Color Histogram." In *Computer Vision and Pattern Recognition*. IEEE, 1992.
- [10] Shafer, S. A. "Using color to separate reflection components." Computer science department, University of Rochester, 1984.
- [11] Tominaga, S. and Wandell, B. A. "Standard surface-reflectance model and illuminant estimation", *J. Opt. Soc. Am.* 6(4), 1989.
- [12] Torrance, K. and Sparrow, E. "Theory for off-specular reflection from roughened surfaces." *J. Opt. Soc. Am.* (57), 1967.

Diffuse And Specular Reflection From Dielectric Surfaces

Lawrence B. Wolff
Computer Vision Laboratory
Department of Computer Science
The Johns Hopkins University
Baltimore, Maryland 21218

Abstract

One of the most common assumptions for recovering object features in computer vision and rendering objects in computer graphics is that the radiance distribution of diffuse reflection from materials is Lambertian. We propose a reflectance model for diffuse reflection from smooth inhomogeneous dielectric surfaces that is empirically shown to be significantly more accurate than the Lambertian model. The resulting reflected diffuse radiance distribution has a simple mathematical form. The proposed model for diffuse reflection utilizes results of radiative transfer theory for subsurface multiple scattering. For an optically smooth surface boundary this subsurface intensity distribution becomes altered by Fresnel attenuation and Snell refraction making it become significantly non-Lambertian. The reflectance model derived in this paper accurately predicts the dependence of diffuse reflection from smooth dielectric surfaces on viewing angle, always falling off to zero as viewing approaches grazing. This model also accurately shows that diffuse reflection falls off faster than predicted by Lambert's law as a function of angle of incidence, particularly as angle of incidence approaches close to 90° . We present diffuse reflection effects near occluding contours of dielectric objects that are strikingly deviant from Lambertian behavior, and yet are precisely explained by our diffuse reflection model. Our proposed diffuse reflection model has the added feature that it explains the physical origin of diffuse albedo which is typically an *ad hoc* scaling coefficient. This can be used to explain the relative strengths of the specular and diffuse reflection components from inhomogeneous dielectric surfaces purely in terms of the physical parameters of the surface itself.

1 INTRODUCTION

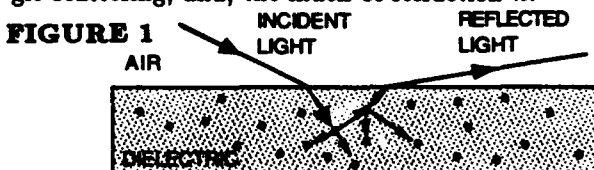
Perhaps the most widely used assumption about reflectance from materials in computer vision and in computer graphics is Lambert's Law for diffuse reflection [7]. Lambert predicted that diffuse reflection from a material contributed by light incident from a specified direction is proportional to the cosine of the angle between this incident direction and the surface normal, independent of the direction of reflection. While relatively little physical motivation was given for this law when it was first published over 200 years ago, it has been adopted by the computer vision and computer graphics communities primarily because it serves as a reasonably accurate and computationally simple approximation for describing diffuse reflection under a number of conditions.

A prevalent class of materials encountered both in common experience and in vision/robotics environments are inhomogeneous dielectrics which include plastics, ceram-

ics, and, rubber. Almost all diffuse reflection from these materials physically arises from subsurface multiple scattering of light caused by subsurface inhomogeneities in index of refraction. In this paper we model inhomogeneous dielectric material as a collection of scatterers contained in a uniform dielectric medium with index of refraction different from that of air. We propose a simple model of diffuse reflected intensity resulting from the process of incident light refracting into the dielectric medium, producing a subsurface diffuse intensity distribution from multiple internal scattering, and then refraction of this subsurface diffuse intensity distribution back out into air. See Figure 1. In [19], [16] we formally derived and empirically verified that if light is incident with radiance, L , at incidence angle, ψ , through a small solid angle, $d\omega$, on a smooth dielectric surface, then

$$\rho L \times (1 - F(\psi, n)) \times \cos \psi \times (1 - F(\sin^{-1}(\frac{\sin \phi}{n}), 1/n)) d\omega$$

describes the diffuse reflected radiance into emittance angle (i.e., viewing angle), ϕ . The terms F refer to the Fresnel reflection coefficients [11], n , is the index of refraction of the dielectric medium, and, ρ , is the total diffuse albedo. We show that the total diffuse albedo, ρ , is directly related to both the single scattering albedo describing the proportion of energy reradiated upon each subsurface single scattering, and, the index of refraction n .



The Lambertian term, $\cos \psi$, embodied in the above expression arises from a combination of subsurface diffuse scattering described by Chandrasekhar's analysis using radiative transfer theory, [2], and a radiometric correction from distortion of infinitesimal angles due to Snell refraction across the air-dielectric boundary. Over ranges of incident and reflected angles where the Fresnel coefficients, F , are insignificantly varying, this offers a formal proof that Lambert's Law does approximate well under these conditions. However, as a result of varying Fresnel coefficients there are a number of conditions where diffuse reflection from smooth inhomogeneous dielectrics seriously deviates from Lambertian behavior.

In the past decade the computer vision and computer graphics communities have become increasingly aware of accurate modeling of the reflectance properties of materials, particularly with respect to the specular component. The works by Torrance and Sparrow [14], and, Beckmann and Spizzichino [1] have been amongst the

most popular in providing computer vision and graphics researchers with an accurate modeling of the specular component of reflection from rough materials [3], [5], [12], [8]. Only very recently has there been consideration of non-Lambertian diffuse reflection within the vision and graphics community. The paper by He and Torrance et al. [4] proposes a comprehensive full-wave model for light reflection including the description of a directional diffuse component produced from diffraction and interference effects when the wavelength of light is comparable to the size of surface roughness elements. This model assumes that diffuse reflection originating from subsurface scattering within inhomogeneous dielectrics is Lambertian and contributes to a uniform diffuse component. Torrance [13] is currently exploring full-wave solutions for light incident on dielectric-air boundaries from subsurface scattering. Tagare and deFigueiredo [12] propose as part of their multiple lobed reflectance model for machine vision a functional approximation to the Chandrasekhar diffuse reflection Law [2]. While we also use the Chandrasekhar diffuse reflection law for diffuse subsurface scattering, it is not nearly accurate for materials without consideration of various dielectric-air boundary effects. Oren and Nayar [9] have been studying non-Lambertian diffuse reflection effects for rough surfaces assuming a statistical distribution of Lambertian reflecting facets along with masking, shadowing, and, interreflection. Apart from analysis of reflected intensity distributions for diffuse reflection, Shafer [10] proposed a *dichromatic* color reflectance model for diffuse and specular components, and Wolff [15] proposed a polarization reflectance model involving the diffuse reflection component for inhomogeneous dielectrics.

This paper is a combined summary of most of the results from the papers [19], [17], [16], [18]. For exact details of the derivations in this paper consult [19], [16], [18].

2 CHANDRASEKHAR DIFFUSE REFLECTION LAW FROM MULTIPLE SCATTERING

Chandrasekhar [2] formally derived a number of expressions for diffuse reflection from multiple scattering under different single scattering conditions. While the original application was to transmission and diffuse reflection of light from stellar and planetary atmospheres, the same physical principles apply to subsurface scattering of light within dielectrics. We are in particular interested in Chandrasekhar's derivation for diffuse reflection assuming that single particle scattering is isotropic. Chandrasekhar assumes that the scattering particles are *plane parallel* in that the optical properties (e.g., particle density) are uniform within parallel planar layers. The geometry of incident and reflected light is referred to with respect to the orientation of this plane of uniformity. In the case of a semi-infinite medium of scatterers (i.e., an opaque dielectric surface) the result for the reflected radiance according to diffuse reflection from multiple scattering, assuming an isotropic single scattering distribution, is expressed in terms of the *Chandrasekhar H-function*:

$$C_\rho(\mu_{inc}, \mu_{ref}) = \frac{\rho}{4\pi} L \frac{\mu_{inc}}{\mu_{ref} + \mu_{inc}} H_\rho(\mu_{inc}) H_\rho(\mu_{ref}),$$

where, μ_{inc} and μ_{ref} are the directional cosines of incident and reflected light with respect to the normal of the plane of uniformity, respectively, ρ is the single scattering albedo (representing the proportion of energy reradiated upon each subsurface single scattering), and L is the incident radiance. The Chandrasekhar H-functions are de-

pendent upon the single scattering albedo, ρ . An N th order approximation to the Chandrasekhar H-function can be expressed;

$$H_\rho(\mu) = \frac{1}{\mu_1 \dots \mu_n} \frac{\prod_{i=1}^N (\mu + \mu_i)}{\prod_{\alpha=1}^N (1 + \kappa_\alpha \mu)},$$

defined in terms of the positive zeros, μ_i , of the even Legendre polynomial of order $2N$, and the positive roots, κ_α , of the associated *characteristic equation*;

$$1 = \sum_{j=1}^N \frac{a_j \rho}{1 - \kappa_j^2 \mu_j^2}.$$

3 DIFFUSE REFLECTION FROM SMOOTH DIELECTRICS

The Chandrasekhar diffuse reflection law alone does not provide a complete physically accurate description of diffuse reflection from inhomogeneous dielectric surfaces. While gaseous molecules of stellar or planetary atmospheres are separated by empty space, particles within an inhomogeneous dielectric surface are assumed to be separated by a uniform medium with index of refraction different from that of air. As we will be considering opaque objects, the assumption of a semi-infinite medium of scatterers is very well approximated. The parallel planes along which the optical properties of subsurface particles are uniform are assumed to be parallel to the smooth planar surface boundary. There are at least 3 ways in which a dielectric medium will alter a subsurface diffuse intensity distribution with respect to its smooth boundary with air. The first 2 ways are a result of Snell's law [11], and the third is due to Fresnel attenuation:

- From Snell's law the angle at which light energy is incident upon (reflected from) the plane of uniformity for the subsurface particles will be different than the angle at which light is incident upon (reflected from) the smooth surface boundary. See Figure 2.
- As a result of Snell's law the solid angle through which light energy is incident upon (reflected from) the plane of uniformity for the subsurface particles will be different than the solid angle through which light is incident upon (reflected from) the smooth surface boundary.
- Because of Fresnel attenuation for transmission of light from air into dielectric and vice versa, light energy incident upon the plane of uniformity for the subsurface particles will be less than light energy originally incident upon the smooth surface boundary, and, light energy transmitted back out through the surface boundary will be less than light energy produced by subsurface diffuse scattering.

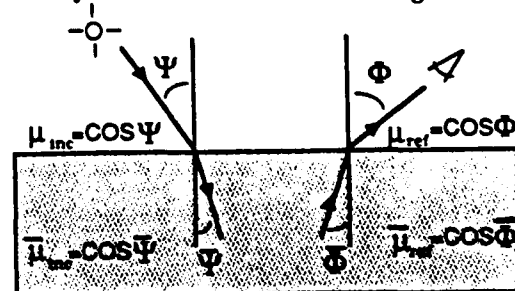


FIGURE 2

Figure 2 defines incident angle ψ and emittance angle ϕ , in air, with respect to the normal of the smooth

dielectric surface boundary. The overline notation refers to the corresponding angles inside the dielectric medium. This correspondence is made according to the following expressions for Snell's law:

$$\bar{\psi} = \sin^{-1}\left(\frac{\sin \psi}{n}\right), \quad \bar{\phi} = \sin^{-1}\left(\frac{\sin \phi}{n}\right) \quad (1)$$

where n is the index of refraction of the dielectric. This is typically in the range of 1.4 to 2.0 for most glasses, plastics, paint and ceramic.

The Fresnel coefficients varying between 0 and 1 inclusive describes the attenuation of incident light radiance upon specular reflection from a planar material interface of size larger than the wavelength of the incident light. The Fresnel coefficients are dependent upon the angle of incidence and the index of refraction of the material. They are also dependent upon the polarization state of light.

The Fresnel coefficient, $F(\psi, n)$ for unpolarized light incident at angle ψ upon a dielectric with simple index of refraction, n , is given by:

$$F(\psi, n) = \frac{1}{2}(F_{\parallel} + F_{\perp}) \quad (2)$$

where

$$F_{\perp}(\psi, n) = \frac{a^2 - 2a \cos \psi + \cos^2 \psi}{a^2 + 2a \cos \psi + \cos^2 \psi}$$

$$F_{\parallel}(\psi, n) = \frac{a^2 - 2a \sin \psi \tan \psi + \sin^2 \psi \tan^2 \psi}{a^2 + 2a \sin \psi \tan \psi + \sin^2 \psi \tan^2 \psi} F_{\perp}(\psi, n)$$

$$a = \sqrt{n^2 - \sin^2 \psi}$$

See Siegal and Howell [11] for a formal derivation.

The details of the derivation of diffuse reflection from smooth dielectric surfaces is given in [19], [16]. It is shown that if light is incident with radiance, L , at incidence angle, ψ , through a small solid angle, $d\omega$, on a smooth dielectric surface, then the reflected radiance is

$$\rho L \times (1 - F(\psi, n)) \times \cos \psi \times (1 - F(\bar{\phi}, 1/n)) d\omega \quad (3)$$

where the total diffuse albedo, ρ , is given by:

$$\rho = \frac{\rho_1}{1 - K} \quad (4)$$

where

$$\rho_1 = \frac{\rho}{n^2} \frac{C_{\rho}(\cos \bar{\psi}, \cos \bar{\phi})}{\cos \bar{\psi}} = \frac{\rho}{4\pi n^2} \frac{H_{\rho}(\bar{\mu}_{inc}) H_{\rho}(\bar{\mu}_{ref})}{\bar{\mu}_{inc} + \bar{\mu}_{ref}}$$

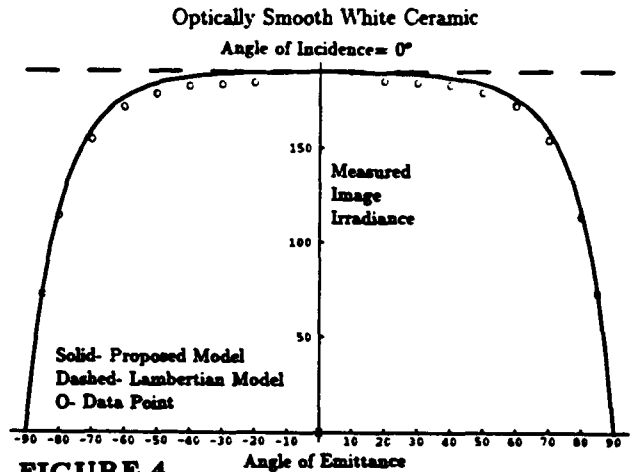
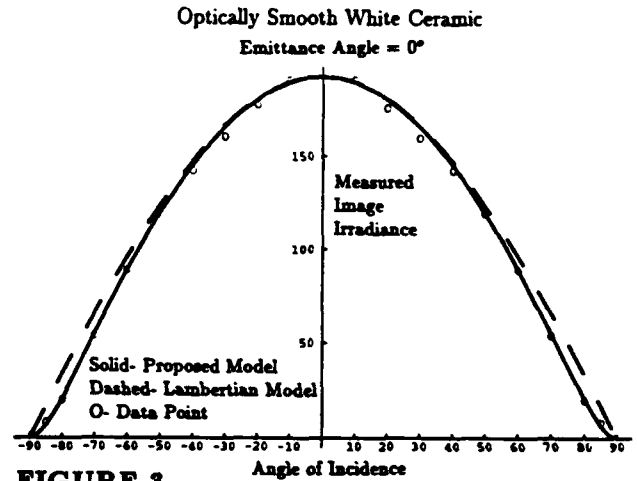
$$K = \int_0^{\pi/2} F(\phi', 1/n) C_{\rho}(\cos \phi', 1.0) 2\pi \sin \phi' d\phi'.$$

It was stated that the derivation of the diffuse reflection formula in this paper includes the physical modeling of diffuse reflection produced from incident light refracting into the dielectric medium (depicted in Figure 1), multiple scattering amongst subsurface inhomogeneities, and refracting back out into air. In fact, this is a first order effect. As light is refracting out into air from the dielectric medium, a significant amount of light is specularly reflected back into the dielectric, particularly for internal specular angles above the critical angle where specular reflection is 100%. This critical angle is $\sin^{-1}(1/n)$ which is typically about 40° for common dielectrics. Light specularly reflected back into the dielectric medium multiply scatters once again, and the term K describes the proportion of light radiance that is internally rescattered. Some of the light radiance that is rescattered refracts from the

dielectric medium into air contributing to diffuse reflection, and some is specularly reflected back into the dielectric medium to be rescattered once again, and so on. The expression for ρ in equation 4 is the result of an infinite geometric series describing the sum of scattering distributions from successive internal specular reflections. It turns out that ρ is only weakly dependent upon imaging geometry (within 3%) and is interpreted as the surface diffuse albedo. References [19], [16] show a plot of total diffuse albedo, ρ , vs. single scattering albedo, ρ .

4 EXPERIMENTAL RESULTS

Figures 3 and 4 show experimental results for an optically smooth piece of compressed white magnesium oxide ceramic. The ceramic was measured with a stylus profilometer to have a variation in height profile no greater than half the wavelength of green light. Using a Brewster



angle technique the index of refraction of the ceramic was determined to be $n = 1.7$. Our diffuse reflectance model enables the empirical measurement of the single scattering albedo, ρ , by empirically determining the ratio of the strengths of the specular and diffuse components of reflection at known angles of incidence and emittance. The ratio of the specular to the diffuse reflection component was measured very near normal incidence and emittance and compared with the ratio in expression 7. The value of ρ can be derived from this, and in turn, ρ can be computed (see [19], [16], [18] for details). The empirically determined specular to diffuse ratio is accounted for by

a single scattering albedo, ρ , just above 0.95 which is remarkably energy conserving.

In Figures 3 and 4, dashed curves represent predicted Lambertian diffuse reflection, solid curves represent the diffuse reflection law of expression 3 with single scattering albedo $\rho = 0.95$ and index of refraction $n = 1.7$ (it makes little difference for $1.0 \geq \rho \geq 0.9$, $1.4 \geq n \geq 2.0$ with respect to the shape of the diffuse reflection curve). Observe how in actuality diffuse reflection goes to zero as viewer angle goes to 90° ! Also, for large angles of incidence percentage error between Lambert's Law and empirical measurements become quite large, exceeding 50% above 80° . The proposed diffuse reflection law, expression 3, never deviates more than 3% from the empirical data while there exist sharp deviations from Lambertian behavior. The results show that the Lambertian model can be assumed to within about 5% accuracy if angle of incidence and angle of emittance are simultaneously no larger than 50° . Outside of this constraint range serious deviations start to occur.

Figure 5 shows an ordinary scene where the Lambertian model strikingly breaks down altogether and yet is explained with high accuracy by the proposed diffuse reflection model in this paper. A ceramic coffee cup of cylindrical body shape is illuminated from the left side by a point source. Starting from the left occluding contour going right the angle of incidence starts at 0° and increases. The Lambertian model predicts that image intensities should decrease going to the right. The image intensities in fact increase to a maximum intensity at about 65° surface orientation and then begin to slowly decrease. The reason is because, $\phi = 90^\circ - \psi$, that is the emittance angle starts at 90° at the occluding contour and decreases going right. Looking at the graph in Figure 4 diffuse reflection increases sharply as angle of emittance decreases from 90° . At about surface orientation 65° ($\psi = 25^\circ$, $\phi = 65^\circ$) the decrease of diffuse reflection with respect to increasing angle of incidence starts to overtake the increase of diffuse reflection with respect to decreasing angle of emittance, and a maximum occurs. (Note the graph in Figure 5 is only for the diffuse component and does not include measurement of the specular component which occurs at relative orientation -45° in the picture of the cup.) According to Figure 5 the qualitative shape of the true diffuse reflection curve (solid) is entirely different (e.g., its not even monotonic) from that for the Lambertian model (dashed). The percentage error of the Lambertian model is also very high for frontal surface orientations where the angle of incidence is large. This behavior of diffuse reflection occurs for a reasonably large range of lighting configurations and can perhaps even aid in the detection of dielectric occluding contours.

Figure 6a shows an actual white billiard ball illuminated by two point light sources orthogonal to viewing, one from the left side and one from the right side. Figure 6b shows a computer graphics rendering of a sphere illuminated by the same configuration of 2 point light sources assuming Lambert's diffuse reflectance Law, while Figure 6c shows the same computer graphics rendering of a sphere using the diffuse reflectance law proposed in this paper. While both shadow boundaries with respect to the left and right light sources coincide along the vertically oriented great circle at the front of the sphere, there appears to be a "shadow band" of darker (i.e., smaller) intensity values about this shadow boundary due to the high fall off of diffuse reflectance at high angles of incidence near 90° . Observe that realistically this "shadow

band" is in fact significantly wider in Figure 6a than predicted by the Lambert Law in Figure 6b. When rendered with the diffuse reflectance law proposed in this paper in Figure 6c, the actual size of the "shadow band" is more accurately predicted. Again, this "shadow band" is not actually a shadow but rather smaller intensity values, and this clearly illustrates the sharper drop off of diffuse intensity values at higher angles of incidence than predicted by Lambert's Law and yet predicted by our diffuse reflectance model (See again Figure 3). The implications for realistic rendering of diffuse reflection in computer graphics is that diffuse reflection produces significantly larger darker regions near shadow boundaries (i.e., at high angles of incidence) than predicted by Lambert's Law. This is particularly true for shadow boundaries occurring at frontal surface orientations relative to the viewer where geometric foreshortening of surface area is not significant.

Figures 7a, 7b, and 7c show grey level representations of isophote curves (i.e., image curves with equal intensity) corresponding respectively to Figures 6a, 6b and 6c. Lambert's Law predicts for this configuration of light sources illuminating a sphere that equal reflected radiance occurs for points forming concentric circles on the sphere about the left-most and right-most occluding contour points. These concentric circles of equal reflected radiance orthographically project onto straight isophote lines as depicted in Figure 7b, with maximum diffuse reflectance occurring at the left-most and right-most occluding contour points where the angle of incidence is zero. Figure 7a which is an actual depiction of the isophotes of Figure 6a shows that in fact lines of equal image intensity severely curve near the occluding contour of the sphere. Maximum diffuse reflection occurs at the center of the closed elliptical isophotes near the left-most and right-most occluding contours, illustrating a 2-D version of the effect depicted in Figure 5. Figure 7c shows the isophotes rendered using the diffuse reflectance model proposed in this paper which are remarkably similar to the actual isophotes in Figure 7a (except for the isophotes perturbed by the specularities). Comparing Figures 7a, 7b and 7c shows very clearly how our diffuse reflectance model accurately predicts reflectance features that are significantly deviant from Lambertian behavior.

5 COMBINED DIFFUSE AND SPECULAR REFLECTION

An important feature of our diffuse reflectance model is that it predicts the surface diffuse albedo for dielectrics purely in terms of physical parameters. Inhomogeneous dielectric surfaces exhibit both diffuse and specular reflection components. Previously, combined diffuse and specular reflection has been modeled as a sum of scaled diffuse and specular terms, with the scaling factors (i.e., *diffuse albedo* and *specular albedo*) determined from experimental fitting for each particular surface. We propose the following combined reflectance model for diffuse and specular reflected radiance from smooth surfaces:

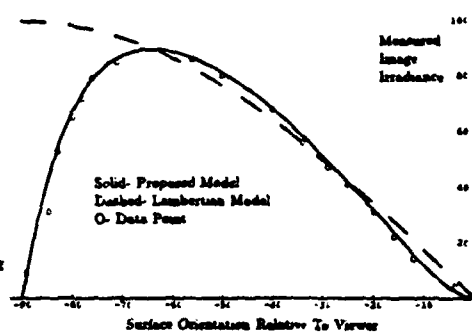
$$L \rho [1 - F(\psi, n)] \cos \psi [1 - F(\sin^{-1}(\frac{\sin \phi}{n}), 1/n)] d\omega + L F(\psi, n) \delta(\psi - \phi) \delta(\theta_0 + 180 - \theta), \quad (5)$$

for small incident solid angle, $d\omega$, at incidence angle ψ , incident azimuth angle θ_0 , emittance angle ϕ and emitted azimuth angle θ . The diffuse albedo, ρ , is that computed from equation 4 and is not just simply a scaling factor but determined from single scattering albedo, ρ , and index of refraction, n .

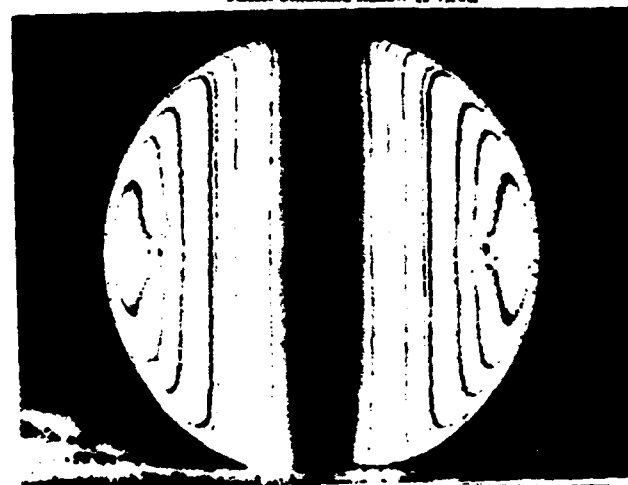


FIGURE 5

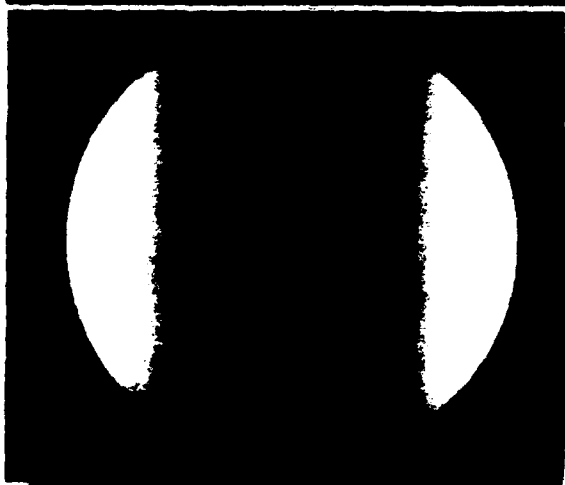
Cylindrical Ceramic Cup
Light Source Incidence Orthogonal to Viewing



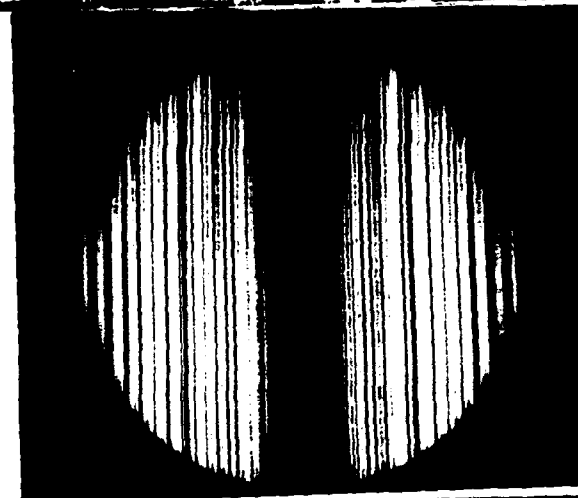
6a



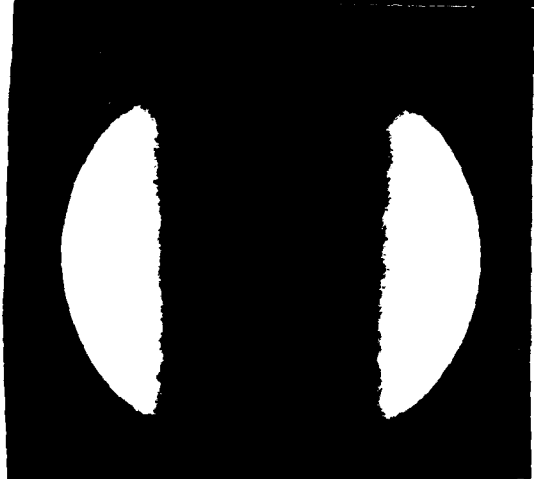
7a



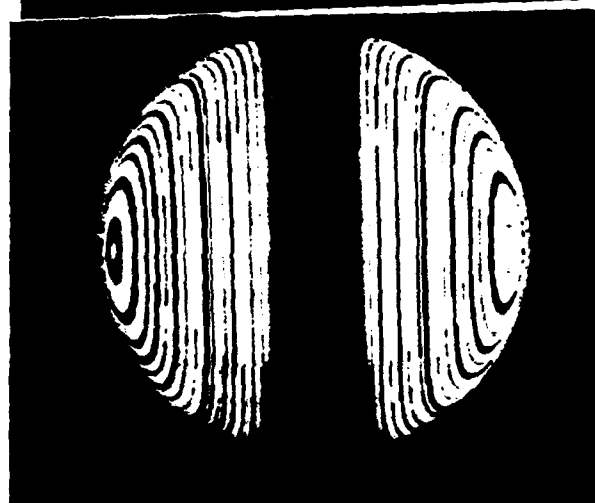
6b



7b



6c



7c

The combined diffuse and specular reflectance model, expression 5 can be used to formally answer the question: How bright is a specularity? Taking the ratio of the strengths of the specular to diffuse reflection components of expression 5:

$$\frac{F(\psi, n)}{\rho[1 - F(\psi, n)] \cos \psi [1 - F(\sin^{-1}(\frac{\sin \phi}{n}), 1/n)] d\omega}, \quad (6)$$

which is independent of uniform incident radiance, L . Since $F(x, n)$ and $F(x, 1/n)$ are monotonically increasing with respect to increasing x it should be clear that diffuse reflection is at a maximum when $\psi = 0^\circ$, $\phi = 0^\circ$ and specular reflection is at a minimum when $\psi = 0^\circ$. Therefore, the ratio expression 6 is a minimum when viewing-illumination geometry is at normal incidence and viewing. The following ratio constitutes a physical lower bound on the ratio of specular to diffuse reflected radiance:

$$\frac{F(0^\circ, n)}{\rho[1 - F(0^\circ, n)][1 - F(0^\circ, 1/n)] d\omega}. \quad (7)$$

The total diffuse albedo, ρ , is largest for conservative scattering when the single scattering albedo $\rho = 1.0$. Using expression 4 to compute ρ at $\rho = 1.0$, the physical lower bound expressed by 7 for the ratio of specular to diffuse reflected radiance evaluated for different indices of refraction is ($d\omega$ in steradians):

$$(n = 1.4) \frac{0.0838}{d\omega} \quad (n = 1.7) \frac{0.211}{d\omega} \quad (n = 2.0) \frac{0.370}{d\omega}.$$

These expressions represent physical lower bounds below which it is not physically possible to have brightness contrast between a specularity and surrounding diffuse reflection on a smooth dielectric surface. Since practically all dielectrics have $n \geq 1.4$, this particular lower bound can serve as a very conservative general formula for ruling out specularities in a scene.

6 CONCLUSION

The primary result of this paper is a simple closed form expression, (equation 3), derived from first physical principles which accurately describes diffuse reflection from a smooth dielectric surface and explains the physical origin of diffuse albedo (equation 4). The results presented in this paper have bearing on virtually any technique in computer vision that relies upon the Lambertian assumption applied to dielectric surfaces, including shape from shading, shape and/or roughness determination from multiple light source illumination (e.g., photometric stereo) and shape from interreflection. It is impossible to reference all of the related works but the book by Horn and Brooks [6] contains a number of applicable papers. The results of this paper makes it possible to analyze the precise conditions under which it is reasonable to assume the Lambertian model for a particular technique, and the conditions under which the Lambertian model breaks down. This more general diffuse reflectance model provides a more solid physical foundation upon which to develop accurate object feature extraction techniques in computer vision. Our model can be used to explain the relative strengths of the specular and diffuse reflection components from smooth inhomogeneous dielectric surfaces purely in terms of the physical parameters of the surface itself [18]. This in turn can be used to describe the relative brightness of specularities on dielectric surfaces.

Acknowledgements

This research was supported in part by an NSF Research Initiation Award, grant IRI-9111973 and DARPA contract F30602-92-C-0191.

References

- [1] P. Beckmann and A. Spizzichino. *The Scattering of Electromagnetic Waves from Rough Surfaces*. Macmillan, 1963.
- [2] S. Chandrasekhar. *Radiative Transfer*. Dover Publications, New York, 1960.
- [3] R. Cook and K. Torrance. A reflectance model for computer graphics. *Journal of Computer Graphics*, 15:307-316, 1981.
- [4] X.D. He, K.E. Torrance, F.X. Sillion, and D.P. Greenberg. A comprehensive physical model for light reflection. *SIGGRAPH Proceedings*, pages 175-186, July 1991.
- [5] G. Healey and T.O. Binford. Local shape from specularly. In *Proceedings of the IEEE first International Conference on Computer Vision (ICCV)*, pages 151-160, London, June 1987.
- [6] B.K.P. Horn and M.J. Brooks. *Shape From Shading*. MIT Press, 1989.
- [7] J. H. Lambert. *Photometria sive de mensura de gratus luminis, colorum et umbrae*. Augsburg, Germany: Eberhard Klett, 1760.
- [8] S. Nayar, K. Ikeuchi, and T. Kanade. Surface reflection: Physical and geometrical perspectives. In *Proceedings of the DARPA Image Understanding Workshop*, pages 185-212, September 1990.
- [9] M. Oren and S. Nayar. *Diffuse Reflection Model for Rough Surfaces*. Columbia University Technical Report CUCS-057-92, November 1992.
- [10] S. Shafer. Using color to separate reflection components. *Color Research and Application*, 10:210-218, 1985.
- [11] R. Siegal and J.R. Howell. *Thermal Radiation Heat Transfer*. McGraw-Hill, 1981.
- [12] H.D. Tagare and R.J.P. deFigueiredo. A theory of photometric stereo for a general class of reflectance maps. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 38-45, San Diego, June 1989.
- [13] K. Torrance. Personal Communication.
- [14] K. Torrance and E. Sparrow. Theory for off-specular reflection from roughened surfaces. *Journal of the Optical Society of America*, 57:1105-1114, 1967.
- [15] L.B. Wolff. *Polarization Methods in Computer Vision*. PhD thesis, Columbia University, January 1991.
- [16] L.B. Wolff. A diffuse reflectance model for dielectric surfaces. In *Proceedings of the SPIE Conference on Optics, Illumination, and Image Sensing for Machine Vision VII*, Boston Massachusetts, November 1992.
- [17] L.B. Wolff. Diffuse reflection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 472-478, Urbana-Champaign Illinois, June 1992.
- [18] L.B. Wolff. *On the Relative Brightness of Specular and Diffuse Reflection*. Johns Hopkins University Technical Report CS 92-30, October 1992 (Submitted to Optical Engineering).
- [19] L.B. Wolff. *A Diffuse Reflectance Model for Dielectric Surfaces*. Johns Hopkins Technical Report CS-92-04, April 1992 (Submitted to the Journal of the Optical Society of America).

Polarization Camera Technology

Lawrence B. Wolff
Computer Vision Laboratory
Department of Computer Science
The Johns Hopkins University
Baltimore, Maryland 21218

Abstract

The more general capabilities of polarisation vision for image understanding motivates the building of camera sensors that automatically sense and process polarisation information. Described in this paper are a variety of designs for *polarization camera* sensors that not only sense partially linearly polarized light, but some of which computationally process polarisation information at pixel resolution to produce a visualisation of sensed polarisation, and/or, a visualisation of physical information directly related to sensed polarization. Described are designs that include the use of radial polarization filters, liquid crystals, beamsplitters, and, VLSI. All designs discussed are currently patent pending.

1 Introduction: Polarization Vision and Polarization Camera Computational Sensors

As human beings we naturally think of vision in terms of perception of intensity and color. Polarisation of light might appear to be of little relevance or benefit to automated vision systems simply because the human visual system is almost completely oblivious to this property of light. In context of Physics-Based Vision there is a compelling motivation to study polarization vision - the extra physical dimensions of polarization, beyond that of intensity of light, carry added information about a world scene that in turn provides a richer set of descriptive physical constraints. As a result, the derivation of some important low-level and high-level descriptions of imaged scenes, which may be infeasible or very difficult to obtain from intensity information alone, can be made possible or can be immensely simplified from analysis of polarization. These include important visual tasks like material classification according to relative electrical conductivity (e.g., dielectric/metal), specular and diffuse reflection component analysis, identification of specular reflection, color constancy, and, image region and image edge segmentations. A detailed description of a variety of polarization-based vision methods are contained in [4], [5], [7], [6], [2].

As Computer Vision algorithms have been designed to perform intensity and color vision so have the video camera sensors (e.g., CCD, CID) that are used in Computer Vision laboratories and for image understanding applications. A criticism that has sometimes been leveled at polarisation-based vision methods is the inconvenience of obtaining polarisation component images by having to place a linear polarizing filter in front of an intensity camera sensor and mechanically rotating this filter by hand or by motor into different orientations. This inconvenience is simply a result of commercially available camera sensors

being geared towards taking intensity images instead of polarization images. In our conception, polarisation vision is no more a "multiple view" problem than is color vision, and a camera sensor can be developed that can automatically sense polarization components and even automatically compute physical scene properties that are directly related to this polarization information. Such a *polarization camera* sensor was originally suggested in [6], and in the past year in the Computer Vision Laboratory at Johns Hopkins we have built and are continuing to develop a variety of designs for such sensors. We discuss in this paper a number of these designs for polarisation cameras that sense and process partially linearly polarized light information.

The *polarization state* of light characterizes its complete description as an electromagnetic wave, apart from wavelength. Polarization is a more general physical description of light than intensity which characterizes its energy. For instance, intensity can be derived from a linear sum of polarization components. Practically all light occurring in robotic/vision environments and naturally occurring light fields is *partially linearly polarized* which means that the polarization state of such light can be represented by the superposition of unpolarized and completely linearly polarized component states (this includes unpolarized and linearly polarized states themselves). A state of partially linearly polarized light can be uniquely measured by sensing light after passing through a *dichroic* material which absorbs all component orientations of polarization except along one axis (i.e., only one axis of polarization is transmitted through the material). If a dichroic filter is made so that all parts of the filter have the same transmission axis (e.g., a standard polarizing filter) then the transmitted radiance of light through the filter as a function of angular orientation of the transmission axis will be a sinusoid with periodicity of 180° as depicted in Figure 1. This sinusoid can be experimentally recovered by taking transmitted radiance measurements for 3 or more unique orientations of the transmission axis. The transmitted radiance sinusoid can be completely described by the parameters I_{max} , I_{min} , and, the *phase*, θ , of the sinusoid which represents its relative horizontal translation in the graph of Figure 1 (e.g., the angular orientation at which I_{min} occurs with respect to 0° on the polariser vernier). Another set of three parameters which characterizes partially linearly polarized light that are of direct importance to polarization-based image understanding are:

$$\begin{aligned} &(\text{partial polarization}) \frac{I_{max} - I_{min}}{I_{max} + I_{min}}, \\ &(\text{total intensity}) I_{max} + I_{min}, \\ &(\text{phase}) \theta. \end{aligned} \quad (1)$$

It can be shown that *partial polarization* represents the fraction between 0 and 1.0 that the linearly polarized component makes up of partially linearly polarized light [3]. The relative phase, θ , represents the angular orientation of the linear component of polarization. Thus partially linearly polarized light provides 2 extra physical dimensions of sensory information beyond intensity of light.

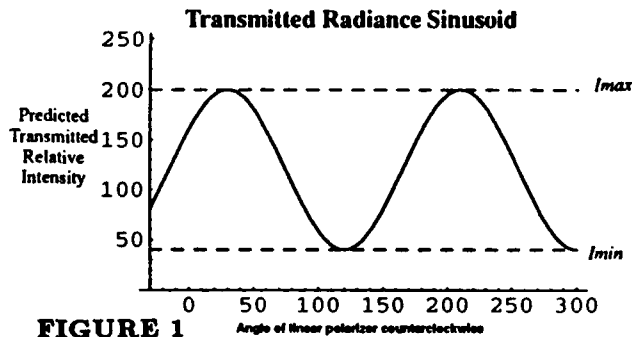


FIGURE 1

The three parameters in equations 1 describing partially linearly polarized light can be recovered by transmitted radiance measurements I_0 , I_{45} , I_{90} taken at 0° , 45° , and 90° angular orientation of the transmission axis, respectively:

$$\theta = (1/2) \tan^{-1} \left(\frac{I_0 + I_{90} - 2I_{45}}{I_{90} - I_0} \right),$$

if ($I_{90} < I_0$) [if ($I_{45} < I_0$) $\theta = \theta + 90$ else $\theta = \theta - 90$]

$$\begin{aligned} I_{max} + I_{min} &= I_0 + I_{90}, \\ \frac{I_{max} - I_{min}}{I_{max} + I_{min}} &= \frac{I_{90} - I_0}{(I_{90} + I_0) \cos 2\theta}. \end{aligned} \quad (2)$$

Of course, 3 other angular orientations could be used, and more than 3 angular orientations can be used to overconstrain the transmitted radiance sinusoid. We have found that the above measurements perform very well.

Visualizing Polarization

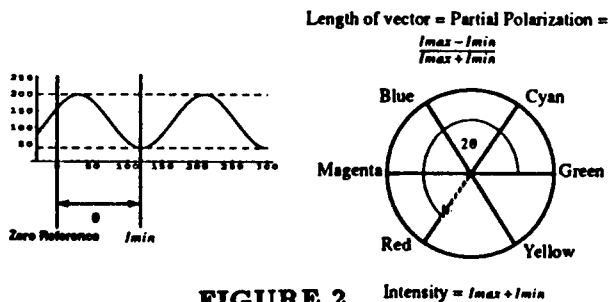


FIGURE 2

Because humans do not observe polarization directly except with the aid of special filters, it is beneficial for a polarization camera to produce some kind of visualization for representing sensed polarization information (e.g., intensity-color representation) for scene analysis. We utilize a hue-saturation-intensity visualization for partially linearly polarized light Wolff and Mancini [8]. Such a scheme was suggested by Bernard and Wehner [1] as a functional similarity between polarization vision and color vision for biological vision systems. Figure 2 shows a natural one-to-one mapping of a state of partial linear polarization into a hue, saturation (i.e., excitation purity), and, intensity, derived respectively from the orientation of the

plane of the completely linear polarized component, the partial polarization, and the intensity of the light. Therefore, in a polarization image, unpolarized light appears achromatic and regions that are significantly partially polarized appear chromatically saturated. The intensity of light in a polarization image is simply the pixel intensity itself, regardless of color, and can be easily processed by intensity-based vision methods. This distinctly demonstrates how a polarization image is a generalization of a gray level intensity image. A number of color polarization images using this visualization are displayed on the last page of this article.

The use of partially linearly polarized light for image understanding is explained in detail in [5], [7], [6], [2], [8]. Only a few main points will be summarized here. Significant partial polarization (i.e., above 10%) in a scene can be due to specular reflection, and/or diffuse reflection from inhomogeneous dielectric objects near occluding contours. The transmitted radiance sinusoids for the specular and diffuse reflection components are respectively 90° out of phase. With respect to the plane determined by the surface normal and the viewing vector, the maximum transmitted radiance for specular reflection occurs for orientation perpendicular to this plane, while for diffuse reflection at occluding contours of inhomogeneous dielectric objects the maximum transmitted radiance occurs for orientation parallel to this plane. This is an important physical principle that can be exploited to help distinguish between partial polarization due to specular reflection and diffuse reflection. On smooth and mildly rough surfaces the phase of the transmitted radiance sinusoid gives surface normal constraint information [4], [6], [7]. The pattern of transmitted radiance sinusoid phases from specular reflection occurring at multiple surface orientations on an object gives physical shape cues that can be exploited for object recognition.

Another important mode of physical information for interpreting objects in a scene is identification of intrinsic material classification. It turns out that if the specular angle of incidence is between 30° and 80° , and the specular component of reflection is strong relative to the diffuse component, the quantity, I_{max}/I_{min} , derived from transmitted radiance sinusoid parameters, is a very reliable discriminator for varying levels of electrical conductivity. This ratio for most metals varies between 1.0 and 2.0 while for dielectrics this ratio is above 3.0. The theory of this is explained in [5], [6].

Whether a polarization camera computes a visualization of sensed polarization information at each pixel, or computes a visualization of physical information (e.g., dielectric/metal composition) at each pixel related to sensed polarization, a polarization camera is inherently a *computational sensor*. It should be fully realized that as intensity is a compression of polarization component information, that a polarization camera can function as a conventional intensity camera, so that intensity vision methods can be implemented by such a camera either alone, or, together with polarization-based vision methods. As intensity-based methods are physical instances of polarization-based methods, a camera sensor geared towards polarization vision does not in any way exclude intensity vision, it only generalizes it providing more physical input to an automated vision system! Adding color sensing capability to a polarization camera makes it possible to sense the complete set of electromagnetic parameters of light incident on the camera.

2 Polarization Camera Using Radial Polarization Filters

Figure 1 depicts the magnitude of light radiance of transmitted partially linearly polarized light through a particular transmission axis of a dichroic material, as this transmission axis is rotated. Most standard polarizing filters are made from a dichroic material with constantly aligned transmission axis across the material. Using a standard polarizing filter, recovery of the transmitted radiance sinusoid at each pixel requires rotation of this axis and serially grabbing at least 3 images respective to unique orientations.

There is a simple way of implementing a "1-big-pixel" polarization camera utilizing a *radial polarization filter* that allows recovery of the transmitted radiance sinusoid in a single image. Figure 3a depicts the concentric circle configuration of transmission axes on such a filter. Assuming the same state of partial linear polarization striking across this filter, the magnitude of transmitted radiance along each circular transmission axis will be a complete sinusoid for every 180° of circular arc. Figure 3b shows linearly polarized light (produced by the square filter) passing through a circular radial polarization filter. The pattern looks something like two pairs of paint brush bristles emanating from the center of the filter. The darkest axis going through the center of the circular filter in Figure 3b is where linear polarized light is being extinguished by tangents to circular transmission axes perpendicular to the orientation of the linearly polarized light. The brightest axis, oriented 90° to the darkest axis, is where the orientation of linear polarized light is parallel to tangents of circular transmission axes.

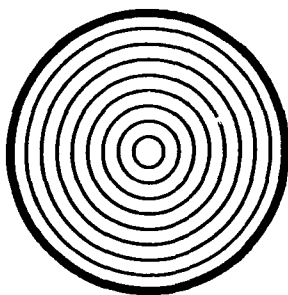


FIGURE 3a

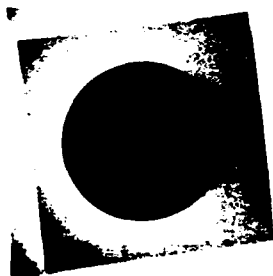


FIGURE 3b

Radial polarization filters are sometimes termed "axis-finders" as the darkest axis tells exactly the orientation of the transmission axis of a linear polarizing filter. However, we are suggesting here that they be used for a more general purpose- for automating the computation of the state of partially linearly polarized light. A radial polarization filter makes it possible to simultaneously measure I_{max} , I_{min} , and the orientation (i.e., phase) at which these occur 90° relative to one another. Clearly, from equations 1 the partial polarization can be easily computed. In fact, the pattern produced by a radial polarization filter is quite an intuitive visualization for partial linear polarization: the contrast between darkest and brightest axes is proportional to the partial polarization (unpolarized light is simply a uniform intensity across the filter), and for non-zero partial polarization the orientation of the darkest axis corresponds to the orientation of the linearly polarized component. It is straightforward to perform image processing operations that will extract this information from a pattern such as the one in Figure 3b across a large set of

pixels. A low cost "1-big-pixel" polarization camera can operate simply by mounting a radial polarization filter on a camera lens and focused on a scene with nearly uniform partial linear polarization within the field of view. At higher cost, multiple radial polarization filters can be utilized for higher resolution measurement of partial linear polarization across a field of view.

3 Polarization Camera Using Liquid Crystals

Obtaining the transmitted radiance sinusoid by rotating a polarizing filter in front of an intensity camera sensor is a mechanically active process that produces optical distortion and is difficult to fully automate. Unless the axis perpendicular to the plane of the polarizing filter is exactly aligned with the optic axis of the camera, small shifts in projection onto the image plane occur between different orientations of the polarizing filter. At intensity discontinuities in a scene, significant shifts in image intensity are observed giving the false interpretation of reflected partial polarization even if it does not exist.



FIGURE 4

Figure 4 shows a liquid crystal polarization camera that has been designed and built at Johns Hopkins using a CCD intensity camera with a fixed polarizer and two Twisted Nematic (TN) liquid crystals mounted in front. The idea behind this liquid crystal polarization camera is very simple. Nothing mechanically rotates; the polarizer remains fixed while the TN liquid crystals electro-optically rotate the plane of the linear polarized component of reflected partially linearly polarized light. The unpolarized component is not effected. In general the transmitted radiance sinusoid can be recovered by the *relative* rotation of the plane of linear polarization with respect to the polarizer. Each TN liquid crystal is binary in the sense that it either rotates the plane of linear polarization by fixed n degrees, $0^\circ < n \leq 90^\circ$, which is determined upon fabrication, and, 0 degrees (i.e., no twist). Two TN liquid crystals are used, one at $n = 45^\circ$, and the other at $n = 90^\circ$, to insure at least 3 samplings of the transmitted radiance sinusoid. Components of partial linear polarization are imaged at pixel resolution under full automatic computer control, and these are processed on a Datacube MV-20 board programmable via Image flow software from a SUN workstation. For details see Wolff and Mancini [8]. One program on the Datacube MV-20 computes from polarization component images a hue-saturation-intensity visualization at each pixel for partial linear polarization. Another program automatically computes dielectric/metal

composition by thresholding I_{max}/I_{min} . Our liquid crystal polarization camera can generate up to 2.5 polarization images a second. The main timing bottle-neck is the relaxation time of 100ms for each of the liquid crystals to switch states. With the most current faster liquid crystals we can at least double the rate of polarization images per second, and we intend to incorporate these newer liquid crystals in our implementation. A nice feature about our liquid crystal polarization camera is that with the Datcube MV-20 board, it is a programmable computational sensor in that sensed polarization components can be processed in a variety of ways.

Color Figures C1, C2, C3, C4, and, C5 at the back of this article show polarization images taken with our liquid crystal polarization camera depicting partial linear polarization at pixel resolution in the hue-saturation-intensity visualization scheme defined in Figure 2.

Figure C1 shows how a polarization image provides important information about a scene that would be very difficult and perhaps impossible to deduce from an intensity image. The left intensity image of Figure C1 shows what apparently are 2 mugs in a scene. Looking closely at the intensity image reveals that there is some difference between the 2 mugs; the left mug has its letters reversed. The only visual cues telling that the left mug is simply a reflection are very high level features such as the reversal of recognizable high level features (e.g., alphabet letters) or the edge of the glass mirror. Otherwise the reflected intensity (and color) of the 2 mugs look essentially the same. This type of problem occurs in vision fairly frequently such as when stray specular glare from objects give the false interpretation that real edges actually exist there. Consider the problem of an autonomous land vehicle viewing a scene part of which is reflected by a lake or river. How does the vehicle know which are the "real" elements of the scene? How does a mobile robot know when it is running into a glass door, or if navigating according to edge cues, which are geometric edge cues opposed to specular edge cues? The right polarization image in Figure C1 shows that the left mug has Cyan chromaticity implying significant partial polarization. Cyan chromaticity is also observed at specular highlights on the right mug as well. (The very bright center of specularities saturate the camera so that pixels record gray level 255 regardless of the state of the TN liquid crystals. This gives a flat transmitted radiance sinusoid, and hence, the appearance of unpolarized light, when in fact the reflected light from these areas are significantly partially polarized. This is a limitation of the dynamic range of the SONY XC-77 CCD camera being used, and NOT our polarization vision algorithm.) Significant partial polarization is also observed at the occluding contour of the right mug as Red color. Note that the hue colors Cyan and Red are complementary colors indicative of transmitted radiance sinusoids 90° out of phase.

Figure C2 shows the intensity and polarization images of a cylindrical cup illuminated with an extended light source so as to produce specular reflection from a number of different surface orientations. The different color hues shown in the polarization image correspond to specular plane surface orientation constraints. In this example, Cyan color hue corresponds to specular planes oriented vertically in the image while the complementary color hue, Red, would correspond to specular planes oriented horizontal in the image. Almost the entire spectrum of color hues is displayed here. Figure C3 shows intensity and polarization images of one hemisphere of a plastic sphere

illuminated with an extended light source. While the polarization image does not give completely unique surface orientation information, the pattern of specular plane constraints gives enough rudimentary shape information to distinguish different shape classes for object recognition. For instance, on a cylindrical shape the lines of constant color hue are parallel to one another (Figure C2) while on a spherical shape lines of constant color hue mutually intersect at a point (Figure C3). Besides being useful in sorting by shape systems in manufacturing, outdoor objects illuminated by skylight serving as an extended illuminator may be able to be distinguished by shape class as well.

Figure C4 shows the polarization image of a white billiard ball under point source illumination. Chromaticity at the occluding contour shows partial polarization of diffuse reflection. According to theory, I_{min} of the transmitted radiance sinusoid occurs for orientation parallel to the occluding contour edge, and the hues correspond to these edge orientations around the ball.

Figure C5 shows the polarization image of a pond surrounded by rocks, grass and trees. Even though skylight is partially polarized, at most angles specular reflection of skylight off of water has I_{min} of the transmitted radiance sinusoid closely aligned with the surface normal to the water, which since it is fluid is aligned with gravity. In this color hue coordinate system, the Green color hue is for I_{min} occurring nearly vertical in this image. While there is noted shift in color hue for ripples in the pond where surface orientation is perturbed, the water has a very distinct reflected polarization signature against the reflected polarization signature of trees and grass which has less chromaticity (i.e., less partial polarization) and variegated color hue (i.e., a wide range of polarization phase). Note also the significant partial polarization from the rocks.

Figure 5a shows a circuit board with solder metal, dielectric, and, solder metal covered with a translucent dielectric material. The circuit board is illuminated with an extended light source so that a strong specular component is reflected from all object points into our liquid crystal polarization camera. Figure 5b shows an image where I_{max}/I_{min} is derived from partial linear polarization at each pixel (and scaled in the range 0-255). Darker regions in Figure 5b represent higher electrical conductivity, lighter regions represent dielectric, intermediate regions are where translucent dielectric covers metal.

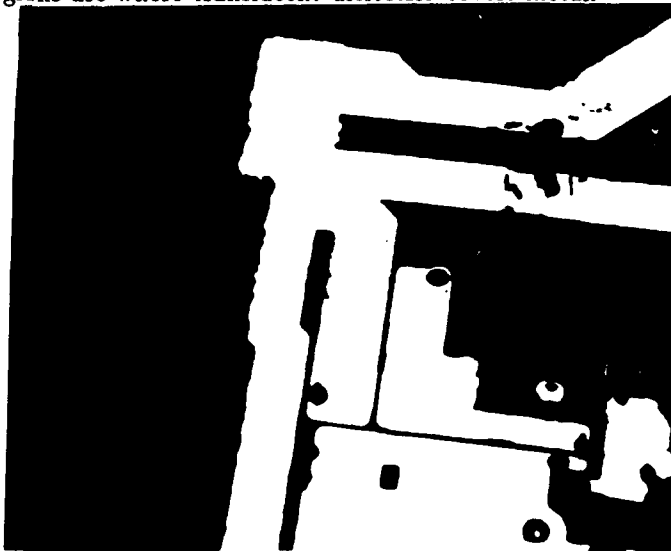


FIGURE 5a



FIGURE 5b

4 Polarization Camera Using Beamsplitter

A common design for high quality color cameras is to use a beamsplitter that directs equal amounts of incoming light onto 3 separate CCD chips for red, green, and blue. A similar idea can be used to direct light onto multiple CCD chips, each chip covered by a uniquely oriented polarizing filter. Unfortunately the polarizing properties of most common kinds of beamsplitters can be variable across standard wide fields of view.

Figure 6 shows a 2-CCD chip polarization camera (i.e., using 2 CCD cameras) utilizing a polarizing plate beamsplitter, built in the Johns Hopkins Computer Vision Laboratory. The simplicity of this design stems from the use of a special coating on a glass plate producing a beamsplitter that effects the polarization of transmitted and reflected light in a nearly constant known way across a fairly wide range of angles (i.e., $\pm 20^\circ$). The polarization state of reflected and transmitted light is effected in a linearly independent way by the plate beamsplitter. If the component of polarization parallel to the floor is represented by P , and the component of polarization vertical to the floor is represented by S , then:

$$aP + bS = I_{\text{transmitted}}$$

$$(1 - a)P + (1 - b)S = I_{\text{reflected}}$$

where $a + b = 1$, $a, b \geq 0$, $a \neq b$. The coefficients a, b are dependent upon the coating on the beamsplitter. This results in the solution

$$S = \frac{I_{\text{transmitted}}(1 - a) - aI_{\text{reflected}}}{b - a},$$

$$P = \frac{I_{\text{transmitted}}(1 - b) - bI_{\text{reflected}}}{a - b}.$$

If the P and S directions happen to coincide with the directions of the maximum and minimum polarization components, or, if the specular plane for specular reflection from an object surface is known, then the partial polarization and phase can be computed (i.e., the transmitted radiance sinusoid can be computed). Otherwise, just the

P and S component magnitudes are known with respect to the mutually orthogonal directions parallel and perpendicular to the floor.

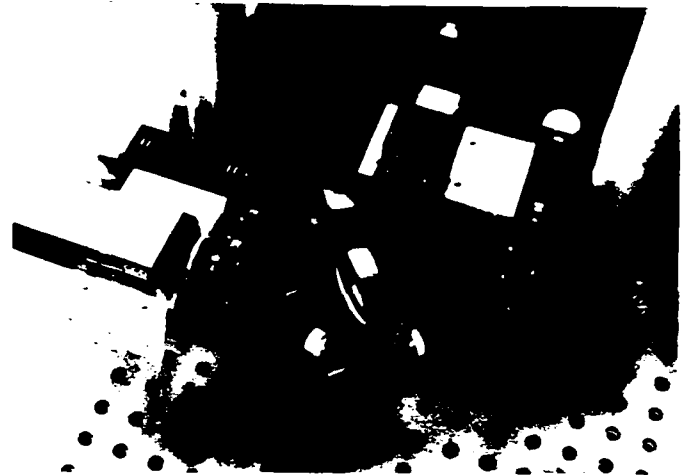


FIGURE 6

By adding a single TN liquid crystal to Figure 5 with the 2 CCD chips and beamsplitter, the P and S components can be measured respective to two mutually orthogonal orientations. With 0 degree twist, the P and S components orientations are parallel and perpendicular to the page, with n degree twist the P and S components are n degree rotations of parallel and perpendicular to the page. As long as n does not equal 90 degrees, the transmitted radiance sinusoid is being sampled in 4 unique points and can be uniquely recovered with any 3 of these points. There are obvious extensions using three CCD chips at the expense of more difficult registration problems.

Polarization information from the 2-CCD polarization camera is processed on our Datacube MV-20 board using the formula

$$\frac{\|P - S\|}{\|P + S\|} \quad (3)$$

In the case where the P and S directions are aligned with the principal polarization component directions (i.e., the directions of I_{\min} and I_{\max}), formula 3 is the partial polarization (equations 1). In general, formula 3 is less than or equal to the partial polarization of light, and even though P and S may not be aligned with the principal directions at a pixel, they can give a measure of "polarization contrast" which can be useful. With the added single TN liquid crystal, the full transmitted radiance sinusoid can be recovered at each pixel, and processing of this polarization information is similar to the way it is done for the liquid crystal polarization camera.

Clearly if $a = b$, there is no solution for the equations solving for P and S above since the simultaneous linear equations to be solved are the same. The case where $a = b$ represents a non-polarizing beamsplitter since the transmitted and reflected beams both have the same polarization state as the incident polarization state, but one half the radiance of the incident beam. The only way polarizing components can be resolved in this case is to place polarizing filters over the CCD chips themselves, each chip having a unique orientation. This can be a feasible design for a polarization camera using a non-polarizing beamsplitter that operates over a sufficiently wide field of view.

5 Polarization Camera Chips

In collaboration with Prof. Andreas Andreou at Johns Hopkins we are in the process of developing self-contained VLSI versions of polarization cameras that sense complete states of partial linear polarization on-chip, compute state of partial linear polarization, and, compute visualization or physical information related to sensed polarization. VLSI offers very high computational throughput so that VLSI polarization cameras can enable operations at very high speeds.

6 Conclusion

We have introduced *polarization camera* computational sensors that:

- Compute sensed polarization from either an image pattern (e.g., radial polarization filters), a sequence of polarization component images (e.g., liquid crystal polarization camera), or, in parallel from multiple polarization component states (e.g., multiple CCD cameras with beamsplitter, self-contained VLSI chip).
- Compute a visualization of polarization information sensed at each pixel (e.g., hue-saturation-intensity representation for partially linearly polarized light).
- Compute a visualization of physical information directly related to the state of sensed polarization at each pixel (e.g., metal/dielectric classification).

We feel that there are considerable advantages to building a polarization camera sensor geared towards doing polarization vision. Polarization cameras have more general capabilities than standard intensity cameras, and there already exist polarization vision methods that can significantly benefit and enhance a number of application areas such as aerial reconnaissance, autonomous navigation (e.g., UGV), target recognition, inspection, and, manufacturing and quality control. Polarization cameras make polarization vision methods more accessible to these application areas and others.

Acknowledgements

The implementations of most of the polarization camera sensors discussed in this paper have been aided by Todd Mancini, Andreas Andreou, and Philippe Pouliquen. This research was supported in part by an NSF Research Initiation Award, grant IRI-9111973 and DARPA contract F30602-92-C-0191.

References

- [1] G. D. Bernard and R. Wehner. Functional similarities between polarization vision and color vision. *Vision Res.*, 17:1019-1028, 1977.
- [2] T.E. Boulton and L.B. Wolff. Physically-based edge labeling. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Maui, June 1991.
- [3] D. Clarke and J.F. Grainger. *Polarized Light and Optical Measurement*. Pergamon Press, 1971.
- [4] L.B. Wolff. Surface orientation from polarization images. In *Proceedings of Optics, Illumination and Image Sensing for Machine Vision II, Volume 850*, pages 110-121, Cambridge, Massachusetts, November 1987. SPIE.
- [5] L.B. Wolff. Polarization-based material classification from specular reflection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 12(11):1059-1071, November 1990.
- [6] L.B. Wolff. *Polarization Methods in Computer Vision*. PhD thesis, Columbia University, January 1991.
- [7] L.B. Wolff and T.E. Boulton. Constraining object features using a polarization reflectance model. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13(7):635-657, July 1991.
- [8] L.B. Wolff and T.A. Mancini. Liquid crystal polarization camera. In *Proceedings of IEEE Workshop on Applications of Computer Vision*, pages 120-127, Palm Springs, California, December 1992.

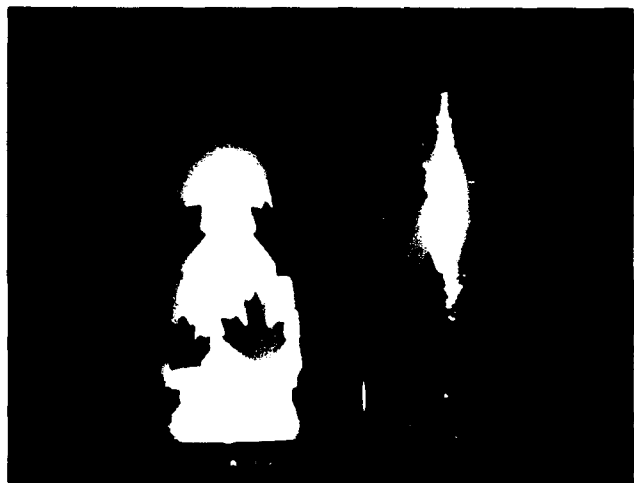


FIGURE C1

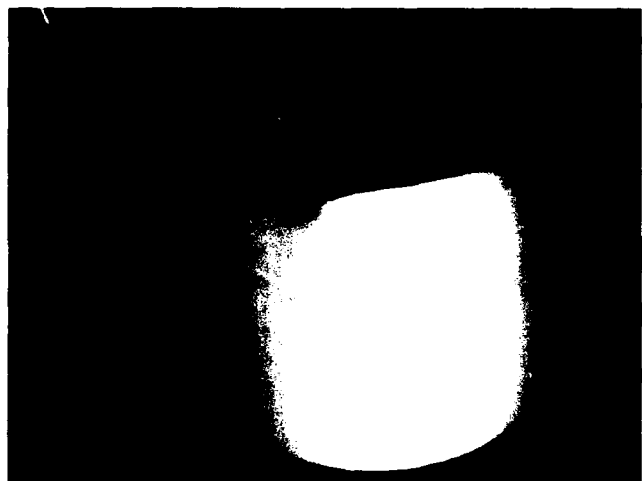


FIGURE C2



FIGURE C3

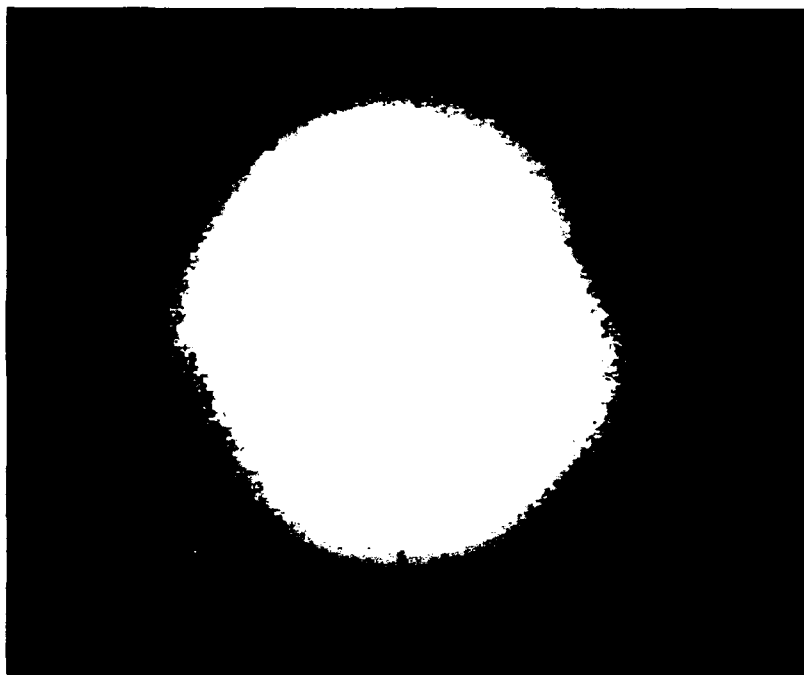


FIGURE C4



FIGURE C5

Generalization of the Lambertian Model

Michael Oren and Shree K. Nayar

Department of Computer Science, Columbia University, New York, N.Y. 10027 *

Abstract

The Lambertian model is one of the most widely used models in machine vision. For several real-world objects, the Lambertian model can prove to be a very inaccurate approximation to the diffuse component. While the brightness of a Lambertian surface is independent of viewing direction, the brightness of a rough diffuse surface increases as the viewing direction approaches the source direction. In this paper, we develop a comprehensive model that predicts reflectance from rough diffuse surfaces. The model accounts for complex geometric and radiometric phenomena such as masking, shadowing, and interreflections between points on the rough surface. The proposed model may be viewed as a generalization of the Lambertian model and describes reflectance characteristics that are not captured by existing models. We have conducted several experiments on samples of rough diffuse surfaces, such as, plaster and sand. All of these surfaces demonstrate significant deviation from Lambertian behavior. The reflectance measurements obtained are in strong agreement with the reflectance predicted by our model. We conclude with a brief discussion on the implications of these results on machine vision.

1 Introduction

Image brightness values are closely related to the reflectance properties of points in the scene. Hence, accurate reflectance models are fundamental to the advancement of machine vision. A surface with Lambertian reflectance appears equally bright from all directions. This model for diffuse reflection is one of the most widely used

assumptions in machine vision. It is used explicitly in the case of shape recovery techniques such as shape from shading and photometric stereo. It is also implicitly used by vision techniques such as binocular stereo and motion detection to solve the correspondence problem.

For several real-world objects, however, the Lambertian model can prove to be a poor and inadequate approximation to the diffuse component. In the areas of machine vision, remote sensing, and computer graphics, each picture element (pixel) can represent a surface area with substantial roughness. Though the Lambertian assumption is often reasonable when looking at a small planar surface element, the roughness of the total surface covered by a pixel causes it to behave in a non-Lambertian manner. This deviation from Lambertian reflectance is significant for very rough surfaces, and increases with the angle of incidence. In this paper, we develop a comprehensive model that predicts reflectance from rough diffuse surfaces, and provide experimental results that support the model. The proposed reflectance model that may be viewed as a vast generalization of the Lambertian model.

Prior to developing the diffuse reflectance model, we conducted a detailed survey of related work in the areas of applied physics and geophysics. Though the topic of rough diffuse surfaces has been extensively studied, a complete model such as the one presented here has not been developed. In 1924, Opik [Opik, 1924] designed an empirical reflectance model to describe the non-Lambertian behavior of the moon. In 1941, Minnaert [Minnaert, 1941] modified Opik's model to obtain the following reflectance function:

$$f_r = \frac{k+1}{2\pi} (\cos \theta_i \cos \theta_r)^{(k-1)} \quad (0 \leq k \leq 1)$$

This function was designed to obey Helmholtz's

*This research was supported in part by DARPA Contract No. DACA 76-92-C-0007 and in part by the NSF Research Initiation Award.

reciprocity principle but is not based on any theoretical foundation. It predicts that the radiance of non-Lambertian diffuse surfaces is symmetrical with respect to the surface normal direction. We will show in this paper that this assumption is incorrect. Hapke and van Horn [Hapke and Horn, 1963] also obtained reflectance measurements from rough diffuse surfaces by varying the source direction for a fixed viewer direction. Their measurements show that the peak of the radiance function is shifted from the peak position expected for a Lambertian surface.

The studies cited above, were attempts to design reflectance models based on measured reflectance data. In contrast, Smith [Smith, 1967] and Buhl et al. [Buhl et al., 1968] attempted to develop a theoretical model for diffuse reflection from rough surfaces. These efforts were motivated primarily by the reflectance characteristics of the moon. Infrared emissions from the moon indicate that the surface of the moon reflects more light back in the direction of the source (the sun) than in the normal direction (like Lambertian surfaces) or in the forward direction (like specular surfaces). This phenomenon is sometimes referred to as *backscattering*¹. Smith modeled the roughness of the moon as a random process and assumed each point on the surface to be Lambertian in reflectance. However, Smith's analysis was confined to the plane of incidence and is not easily extendable to reflections outside the plane of incidence. Buhl et al. [Buhl et al., 1968] modeled the surface as a collection of spherical cavities. They accounted for interreflections using this surface model, but did not present a complete reflectance model that accounts for masking and shadowing effects for arbitrary angles of reflection and incidence.

This paper presents a general and complete model for diffuse reflectance. This model can be applied to isotropic as well as anisotropic rough surfaces, and can handle arbitrary source and viewer directions. Further, it takes into account complex geometrical effects such as *masking*, *shadowing*, and *interreflections* between points on the rough surface. We begin by modeling the surface as a collection of long symmetric V-

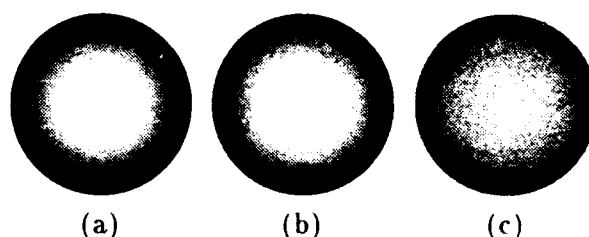


Figure 1: Images of spheres rendered using the proposed reflectance model for rough surfaces: (a) Lambertian sphere ($\sigma = 0$); (b) rough Lambertian sphere ($\sigma = 30^\circ$) (c) rough Lambertian sphere ($\sigma = 70^\circ$).

cavities. Each V-cavity has two opposing facets and each facet is assumed to be much larger than the wavelength of incident. This surface model was used by Torrance and Sparrow [Torrance and Sparrow, 1967] to describe specular reflection from rough surfaces. Here, we assume the facets are Lambertian in reflectance². First, we develop a reflectance model for anisotropic surfaces with one type (facet-slope) of V-cavities, and with all cavities aligned in the same orientation on the surface plane. This result is then used to develop a reflectance model for the more general case of isotropic surfaces that have Gaussian facet-slope distributions with zero mean ($\mu = 0$) and arbitrary standard deviation (σ). The standard deviation represents the macroscopic roughness of the surface. Figure 1 shows three images of spheres rendered using the proposed reflectance model. In all three cases, the sphere is illuminated from the viewer direction. In the first case, $\sigma = 0$, and hence the sphere is Lambertian in reflectance. As the roughness value increases, the sphere begins to appear flatter. In the extreme roughness case shown in Figure 1c, the sphere looks like a flat disc with near constant brightness. This phenomenon has been observed and reported in the case of the full moon.

We present several experimental results that demonstrate the accuracy of our diffuse reflectance model. The experiments were conducted on real samples such as wall plaster and sand. In all cases, the reflectance predicted by the model was found to be in strong agreement with the measurements. These results illustrate

¹A different backscattering mechanism produces a sharp peak close to the source direction (see [Hapke and Horn, 1963, Oetking, 1966, Shibata et al., 1981, Tagare and deFigueiredo, 1991]). This is not the mechanism discussed in this paper.

²In cases where the facets have a specular component in addition to the diffuse component, the model presented in this paper can be used in conjunction with the Torrance-Sparrow model [Torrance and Sparrow, 1967].

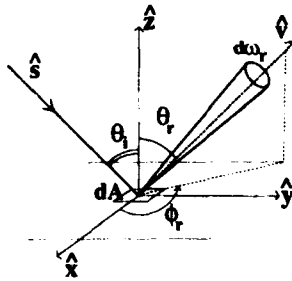


Figure 2: Geometry used to define radiometric terms.

that the deviation from Lambertian behavior can be very significant. For example, when the angle of incidence is 75° , the brightness can vary by a factor of three when the viewing direction is varied. The results presented in this paper demonstrate two points that are fundamental to computer vision. (a) Several real-world objects have diffuse components that are significantly non-Lambertian. (b) The reflectance of such objects cannot be accurately described using any of the previous reflectance models.

2 Radiometric Definitions

In this section, we define radiometric concepts that will be extensively used in the remaining of this paper. These concepts are discussed in detail in [Nicodemus *et al.*, 1977]. Figure 2 shows a surface element dA illuminated from the direction $\hat{s} = (\theta_i, \phi_i)$ and viewed by a sensor (image pixel, for example) in the direction $\hat{v} = (\theta_r, \phi_r)$. We use θ to denote zenith angles and ϕ to denote azimuth angles. The sensor subtends an infinitesimal solid angle $d\omega_r$ from any point on the surface.

The light energy reflected by the surface patch is proportional to the light incident on the patch. *Irradiance* is defined as the light flux incident per unit area of the surface:

$$E(\theta_i, \phi_i) = \frac{d\phi_i(\theta_i, \phi_i)}{dA} \quad (1)$$

This is the directional irradiance of the surface as it represents the light energy incident from the direction (θ_i, ϕ_i) . The brightness measured by the sensor is proportional to the *radiance* of the surface patch in the direction (θ_r, ϕ_r) . Surface radiance is defined as:

$$L_r(\theta_r, \phi_r; \theta_i, \phi_i) = \frac{d\phi_r(\theta_r, \phi_r; \theta_i, \phi_i)}{dA \cos \theta_r d\omega_r} \quad (2)$$

It is the flux radiated by the surface per unit solid angle, per unit *foreshortened* area. It depends on the direction of illumination and the direction of the sensor. The relationship between the irradiance and radiance of the surface is determined by its reflectance properties. The *bi-directional reflectance function (BRDF)* is defined as the ratio of the radiance to the irradiance:

$$f_r(\theta_r, \phi_r; \theta_i, \phi_i) = \frac{dL_r(\theta_r, \phi_r; \theta_i, \phi_i)}{dE(\theta_i, \phi_i)} \quad (3)$$

All of the above definitions are general, in that, they are valid for surfaces with any reflectance characteristics. A special type of reflectance that is widely discussed in computer vision is *Lambertian* reflectance. A Lambertian surface is an ideal diffuser whose radiance is independent of the viewing direction of the sensor; it appears equally bright from all directions. Its BRDF is $f_r = \frac{\rho}{\pi}$ where the constant ρ is the *albedo* of the surface.

3 Surface Roughness Model

All surface models found in applied physics and geophysics literature can be divided into two broad categories. In the first case, the surface is modeled as a random process (see [Beckmann, 1965, Wagner, 1966, Smith, 1967]). Using this approach, it is difficult to derive a reflectance model for arbitrary source and viewer directions as well as to analyze interreflections. In the second category, surfaces are assumed to be composed of several elements with some primitive shape, for example, spherical cavities, V-cavities, holes, etc (see [Buhl *et al.*, 1968, Torrance and Sparrow, 1967]). As we show in this paper, the effects of shadowing, masking, and interreflection need to be modeled in order to obtain an accurate reflectance model. To accomplish this, we use the roughness model proposed by Torrance and Sparrow [Torrance and Sparrow, 1967] that assumes the surface to be composed of long symmetric V-cavities (see Figure 3). Each cavity is composed of two facets. The width of each facet is assumed to be small compared to its length. The roughness of the surface is determined by a probability function used to model the distribution of facet slopes.

We assume each facet area da is small compared to the area dA of the surface patch that is imaged

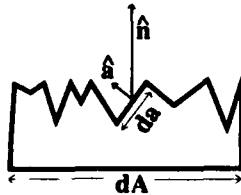


Figure 3: Surface modeled as a collection of V-cavities.

by a single pixel. Hence, each pixel includes a very large number of facets. Further, the facet area is large compared to the wavelength λ of incident light and hence geometrical optics can be used to derive the reflectance model. The above assumptions can be summarized as: $\lambda^2 \ll da \ll dA$. The facets could be relatively small as in the case of sand and plaster, or large as in the case of outdoor scenes of terrain.

Slope-Area Probability Distribution:

We denote the slope and orientation of each facet in the V-cavity model as (θ_a, ϕ_a) . Torrance and Sparrow have assumed all facets to have equal area da . They use the distribution $n(\theta_a, \phi_a)$ to represent the number of facets per unit surface area that have the normal $\hat{a} = (\theta_a, \phi_a)$. Here, we use a probability distribution to represent the fraction of the surface area that includes facets of a particular slope. We refer to this as the *slope-area distribution* $P(\theta_a, \phi_a)$. The facet number distribution and the slope-area distribution can be related as follows:

$$\begin{aligned} P(\theta_a, \phi_a) &= \frac{dA n(\theta_a, \phi_a) da \cos \theta_a}{dA} \\ &= n(\theta_a, \phi_a) da \cos \theta_a \end{aligned} \quad (4)$$

The slope-area probability distribution is easier to use than the facet-number distribution in the following derivation of the reflectance model. For isotropic surfaces, we simply have $n(\theta_a, \phi_a) = n(\theta_a)$ and $P(\theta_a, \phi_a) = P(\theta_a)$ since the distributions are rotationally symmetric with respect to the global surface normal.

4 Reflectance Model

In this section, we derive a reflectance model for rough diffuse surfaces. The V-cavity model is used to describe the surface geometry and each facet on the surface is assumed to be Lambertian in reflectance. The following three types of

surfaces with different slope-area probability distributions are examined. (a) **Uni-directional Single-Slope Distribution:** This distribution results in a non-isotropic surface where all facets have the same slope and all cavities are aligned in the same direction. (b) **Isotropic Single-Slope Distribution:** Here, all facets have the same slope but they are uniformly distributed in orientation on the surface. (c) **Gaussian Distribution:** This is the most general case examined where the slope-area distribution is assumed to be normal with mean zero. The roughness of the surface is determined by the standard deviation of the normal distribution. The reflectance model obtained for each of the above surface types is used to derive the succeeding ones.

The Projected Radiance:

Consider a surface area dA that is imaged by a single sensor element in the direction $\hat{v} = (\theta_r, \phi_r)$, and illuminated by a distant point light source in the direction $\hat{s} = (\theta_i, \phi_i)$. The area is composed of a very large number of symmetric V-cavities. Each V-cavity is composed of two facets with the same slope but facing in opposite directions. Our approach is to compute the radiance contribution of each facet on the surface. Then, the total radiance of the surface patch can be computed as an aggregate of the contributions of all facets. Consider the flux reflected by a facet with area da and normal $\hat{a} = (\theta_a, \phi_a)$. As shown in Figure 3, the projected area on the surface occupied by the facet is $da \cos \theta_a$. Hence, while computing the contribution of the facet to the radiance of the surface patch, we need to use the projected area $da \cos \theta_a$ and not the actual facet area da . The radiance contribution of the facet thus determined is what we call the *projected radiance* of the facet and is given by:

$$L_{rp}(\theta_a, \phi_a) = \frac{d\Phi_r(\theta_a, \phi_a)}{(da \cos \theta_a) \cos \theta_r d\omega_r} \quad (5)$$

For ease of description, we have dropped the source and viewing directions from the notations for radiance and flux in the above expression.

Total Radiance:

Now consider the slope-area distribution of facet orientations given by $P(\theta_a, \phi_a)$. The total radiance of the surface can be obtained as the average of the projected radiance $L_{rp}(\theta_a, \phi_a)$ of all

the facets on the surface:

$$L_r = \int_{\theta_a=0}^{\frac{\pi}{2}} \int_{\phi_a=0}^{2\pi} P(\theta_a, \phi_a) L_{rp}(\theta_a, \phi_a) d\phi_a d\theta_a \quad (6)$$

4.1 Model for Uni-directional Single-Slope Distribution

The first surface type we consider has all facets with the same slope θ_a . Further, all the V-cavities are aligned in the same direction. The results obtained for this anisotropic surface geometry will later be used in the analysis of isotropic surfaces.

Radiance from a Lambertian Facet:

Consider a Lambertian facet that is fully illuminated (no shadowing, and is completely visible (no masking) from the sensor direction. The radiance of the facet is proportional to its irradiance and is given by:

$$L_r(\theta_a, \phi_a) = \frac{\rho}{\pi} E(\theta_a, \phi_a) \quad (7)$$

The irradiance of the surface is $E(\theta_a, \phi_a) = E_0 \langle \hat{s}, \hat{a} \rangle$, where E_0 is the irradiance when the surface is illuminated head-on. Using the definition of radiance, the flux reflected by the facet in the sensor direction is obtained as:

$$d\Phi_r = \frac{\rho}{\pi} E_0 \langle \hat{s}, \hat{a} \rangle \langle \hat{v}, \hat{a} \rangle da d\omega_r \quad (8)$$

Substituting the above expression for reflected flux in equation 5, we obtain the projected radiance for the facet:

$$L_{rp}(\theta_a, \phi_a) = \frac{\rho}{\pi} E_0 \cos \theta_i \cos \theta_a \left(1 + \tan \theta_s \tan \theta_a \cos(\phi_a - \phi_i) \right) \left(1 + \tan \theta_v \tan \theta_a \cos(\phi_a - \phi_r) \right) \quad (9)$$

This expression clearly indicates that the projected radiance of a tilted Lambertian facet is not equal in all viewing directions. Hence, a rough Lambertian surface comprised of tilted facets reflects in a non-Lambertian manner; its radiance varies with the viewing direction. *This phenomenon is observed even in the absence of masking, shadowing, and interreflection effects.*

4.1.1 Geometric Attenuation Factor

If the surface is illuminated and viewed head-on, all of the facets are fully illuminated and visible. For larger angles of incidence and reflection, however, facets are shadowed and masked

by adjacent facets (see Figure 4). In the case of *shadowing*, a facet is only partially illuminated as the adjacent facet on the V-cavity casts a shadow on it. In the case of *masking*, the facet is only partially visible to the sensor as its adjacent facet occludes it. Both these geometrical phenomena affect the projected radiance of the facet and hence must be taken into account. Torrance and Sparrow [Torrance and Sparrow, 1967] have studied these effects while deriving their reflectance model for specular facets. We analyze the same effects here for Lambertian facets. The result is a *geometrical attenuation factor* (\mathcal{GAF}) that lies between zero and unity. It represents the reduction in the projected radiance of a facet due to masking and shadowing effects. Consider

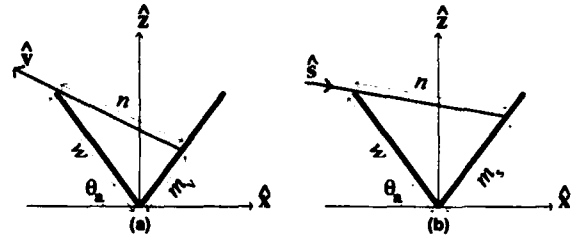


Figure 4: Masking and shadowing effects.

the masked and shadowed regions shown in Figure 4. If the visible area of a facet is smaller than the illuminated area, the masking effect dominates. On the other hand, if the illuminated area is smaller than the visible area, the shadowing effect dominates. We denote the length and width of the facet by l and w , respectively. Further, m_s and m_v are the sections of the facet that are shadowed and masked, respectively. The area of the facet which is both illuminated and visible is $l \cdot \text{Min}[w - m_s, w - m_v]$. The \mathcal{GAF} is obtained by dividing this expression by the area wl of the facet:

$$\mathcal{GAF} = \text{Min} \left[1 - \frac{m_s}{w}, 1 - \frac{m_v}{w} \right] \quad (10)$$

We need to express the \mathcal{GAF} in terms of the source direction (θ_i, ϕ_i) and the sensor direction (θ_r, ϕ_r) . For any given set of source and sensor angles, the shadowed and masked regions m_s and m_v can be derived using trigonometry. This derivation has already been presented in [Blinn, 1977] and hence we will not discuss it here. The final expression for the \mathcal{GAF} is found to be:

$$\mathcal{GAF} = \text{Min} \left[1, \text{Max} \left[0, \frac{2 \langle \hat{n}, \hat{a} \rangle \langle \hat{n}, \hat{s} \rangle}{\langle \hat{a}, \hat{s} \rangle}, \frac{2 \langle \hat{n}, \hat{a} \rangle \langle \hat{n}, \hat{v} \rangle}{\langle \hat{a}, \hat{v} \rangle} \right] \right] \quad (11)$$

Projected Radiance and \mathcal{GAF} :

The projected radiance of a Lambertian facet is obtained by simply multiplying the geometric attenuation factor with the projected radiance obtained under the assumption of no masking and shadowing effects. Table 1 details the \mathcal{GAF} and the corresponding projected radiance for all cases of shadowing and masking. Note that the projected radiance is denoted as L_{rp}^1 ; the superscript 1 is used to indicate that the radiance is due to direct illumination by the source. In the next section, we will use L_{rp}^2 to denote the radiance due to interreflections (multiple reflections).

4.1.2 Interreflection Factor

In our reflectance model, we also account for interreflections; light rays bouncing between adjacent facets. These effects are significant for rough surfaces with relatively high albedo values. We have the task of modeling interreflections in the presence of masking and shadowing effects. In the case of Lambertian surfaces, the energy in an incident light ray diminishes rapidly with each interreflection bounce. Hence, we model only two-bounce interreflections and ignore subsequent bounces. The experimental results show that this approximation is a good one. Interreflections are more significant for very rough surfaces and less so for surfaces with mostly "open" V-cavities. Since the long V-cavity model is only an approximation to the real surface, we propose to use a parameter χ , where $0 < \chi \leq 1$, to represent the coefficient of the interreflection component of radiance. The real surface may not have very long cavities and hence the parameter χ can be adjusted to account for such discrepancies. For very rough surface with deep cavities, $\chi \approx 1$, and for relatively open and shorter cavities (as in the case of terrain), $\chi \approx 0.5 - 0.75$.

In the following discussion, we refer to surface radiance due to direct illumination by the source as L_r^1 and the radiance due to the second bounce (first interreflection) as L_r^2 . We will use the same superscripts for the projected radiance. The two-bounce interreflection component for a Lambertian facet can be expressed as [Koenderink

and van Doorn, 1983]:

$$L_r^2(\vec{x}) = \frac{\rho}{\pi} \iint K(\vec{x}, \vec{y}) L_r^1(\vec{y}) d\vec{y} \quad (12)$$

where \vec{x} is a point on the facet whose interreflection component is determined as an integral of the radiance of all points \vec{y} on the adjacent facet. $K(\vec{x}, \vec{y})$ is called the *kernel* and represents the geometrical relationship between \vec{x} and \vec{y} . Since the V-cavity is very long compared to its width, it can be viewed as a two-dimensional shape with translational symmetry. For such shapes, the interreflection component can be determined as an integral over the two-dimensional cross-section of the shape. The above interreflection equation is therefore reduced to:

$$L_r^2(x) = \frac{\rho}{\pi} \int K'(x, y) L_r^1(y) dy \quad (13)$$

where x and y are the shortest distances of points \vec{x} and \vec{y} from the intersection of the two facets (see Figure 5). K' is the kernel for the translational symmetry case and is derived in [Forsyth and Zisserman, 1989] to be:

$$K'(x, y) = \frac{\pi \sin^2(2\theta_a)}{2} \frac{xy}{(x^2 + 2xy \cos(2\theta_a) + y^2)^{3/2}} \quad (14)$$

We know that the orientation of the considered

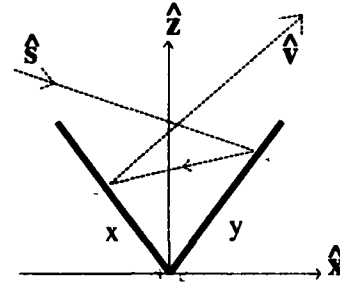


Figure 5: Interreflections in a V-cavity.

facet is $\hat{a} = (\theta_a, \phi_a)$ and the orientation of the adjacent facet is $\hat{a}' = (\theta_a, \phi_a + \pi)$. The limits of the integral in the interreflection equation are determined by the masking and shadowing of the facets. Let m_v be the width of the facet which is visible to the viewer. Let m^s be the width of the adjacent facet that is illuminated. As in section 4.1.1, expressions can be obtained for the visible and illuminated sections:

$$\frac{m_v}{w} = \text{Max} \left[0, \text{Min} \left[1, -\frac{\langle \hat{a}', \hat{v} \rangle}{\langle \hat{a}, \hat{v} \rangle} \right] \right] \quad (15)$$

$$\frac{m^s}{w} = \text{Max} \left[0, \text{Min} \left[1, -\frac{\langle \hat{a}, \hat{s} \rangle}{\langle \hat{a}', \hat{s} \rangle} \right] \right] \quad (16)$$

	GAF	$L_{rp}^1(\theta_a, \phi_a)$
No Masking or Shadowing	1	$\frac{\rho}{\pi} E_0 \frac{\langle \hat{a}, \hat{s} \rangle \langle \hat{a}, \hat{v} \rangle}{\langle \hat{n}, \hat{a} \rangle \langle \hat{n}, \hat{v} \rangle} =$ $\frac{\rho}{\pi} E_0 \cos \theta_i \cos \theta_a (1 + \tan \theta_a \tan \theta_i \cos(\phi_a - \phi_i))$ $(1 + \tan \theta_a \tan \theta_r \cos(\phi_a - \phi_r))$
Masking	$\frac{2\langle \hat{n}, \hat{v} \rangle \langle \hat{n}, \hat{a} \rangle}{\langle \hat{v}, \hat{a} \rangle}$	$\frac{\rho}{\pi} E_0 2\langle \hat{a}, \hat{s} \rangle =$ $\frac{\rho}{\pi} E_0 \cos \theta_i \cos \theta_a 2(1 + \tan \theta_a \tan \theta_i \cos(\phi_a - \phi_i))$
Shadowing	$\frac{2\langle \hat{n}, \hat{s} \rangle \langle \hat{n}, \hat{a} \rangle}{\langle \hat{s}, \hat{a} \rangle}$	$\frac{\rho}{\pi} E_0 \frac{2\langle \hat{n}, \hat{s} \rangle \langle \hat{n}, \hat{a} \rangle}{\langle \hat{v}, \hat{n} \rangle} =$ $\frac{\rho}{\pi} E_0 \cos \theta_i \cos \theta_a 2(1 + \tan \theta_a \tan \theta_r \cos(\phi_a - \phi_r))$

Table 1: Projected radiance of a facet for different masking/shadowing conditions.

By using the following change of variables: $r = \frac{x}{w}$; $t = \frac{y}{w}$, we get the projected radiance due to two-bounce interreflections as:

$$L_{rp}^2 = \left(\frac{\rho}{\pi}\right)^2 E_0 \frac{\langle \hat{a}', \hat{s} \rangle \langle \hat{a}, \hat{v} \rangle}{\langle \hat{a}, \hat{n} \rangle \langle \hat{v}, \hat{n} \rangle} \int_{t=\frac{m_v}{w}}^1 \int_{r=\frac{m_s}{w}}^1 K'(r, t) dr dt \quad (17)$$

Using equation 14, the integral is evaluated as:

$$\int_{t=\frac{m_v}{w}}^1 \int_{r=\frac{m_s}{w}}^1 K'(r, t) dr dt = \quad (18)$$

$$\frac{\pi}{2} \left[d\left(1, \frac{m_v}{w}\right) + d\left(1, \frac{m_s}{w}\right) - d\left(\frac{m_s}{w}, \frac{m_v}{w}\right) - d(1, 1) \right]$$

where:

$$d(x, y) = \sqrt{x^2 + 2xy \cos(2\theta_a) + y^2} \quad (19)$$

We refer to the above term as the *interreflection factor* (\mathcal{IF}). Then, the interreflection component of the projected radiance of a facet with orientation (θ_a, ϕ_a) can be written as:

$$L_{rp}^2 = \left(\frac{\rho}{\pi}\right)^2 E_0 \cos \theta_i \cos \theta_a$$

$$(1 - \tan \theta_a \tan \theta_s \cos(\phi_a - \phi_i)) \quad (20)$$

$$(1 + \tan \theta_a \tan \theta_v \cos(\phi_a - \phi_r)) \mathcal{IF}(\hat{v}, \hat{s}, \hat{a})$$

The total projected radiance of the facet is the sum of the projected radiance due to source illumination (given in Table 1) and the interreflection component:

$$L_{rp}(\theta_a, \phi_a) = L_{rp}^1(\theta_a, \phi_a) + \chi L_{rp}^2(\theta_a, \phi_a) \quad (21)$$

As discussed earlier in this section, we use the parameter χ to weight the contribution due to interreflections to account for discrepancies in the

interreflections predicted by the V-cavity surface model and the real surface. Note that the unidirectional single-slope surface we have considered in this section has only two types of facets with normals (θ_a, ϕ_a) and $(\theta_a, \phi_a + \pi)$. Hence, the radiance of the surface for any given source direction and sensor direction is simply the average of the projected radiance of the two facet types:

$$L_r = \frac{L_{rp}(\theta_a, \phi_a) + L_{rp}(\theta_a, \phi_a + \pi)}{2} \quad (22)$$

4.2 Model for Isotropic Single-Slope Distribution

We now consider a surface where all V-cavities have facets with the same slope (θ_a) , but the V-cavities are uniformly distributed in orientation (ϕ_a) in the plane of the surface. The result is a surface with isotropic roughness. The reflectance model derived for this surface is based on the results obtained for the single slope surface in the previous section. The results obtained in this section are important as they can be used to obtain a reflectance model for any isotropic surface.

Consider a surface with only one type of V-cavities that are distributed uniformly in the plane of the surface. From the previous section, we know the projected radiance $L_{rp}^1(\theta_a, \phi_a)$ of a facet with normal $\hat{a} = (\theta_a, \phi_a)$. All facets on the isotropic surface have the same slope θ_a but different orientation ϕ_a . Hence, the radiance of the isotropic surface is obtained as an integral of the projected radiance over ϕ_a :

$$L_{rp}^1(\theta_a) = \frac{1}{2\pi} \int_{\phi_a=0}^{2\pi} L_{rp}^1(\theta_a, \phi_a) d\phi_a \quad (23)$$

For lack of space, we will avoid detailed derivations and focus on the main results obtained.

We approach the problem as follows: Given a source direction (θ_i, ϕ_i) and a sensor direction (θ_r, ϕ_r) we need to find the ranges of facet orientation ϕ_a for which the facets are masked, shadowed, masked and shadowed, and neither masked nor shadowed. The projected radiance for each range is given in Table 1. The problem then is to decompose the above integral into different parts each corresponding to a different masking/shadowing range³. Using basic geometry, we have identified the limits of the integrals corresponding to different ranges of shadowing/masking. These limits are represented by the *critical angles* ϕ_c^s (for shadowing) and ϕ_c^v (for masking). The critical angle ϕ_c^s is related to the slope θ_a of all surface facets as follows:

$$\phi_c^s = \begin{cases} \cos^{-1} \left(\frac{1}{\tan \theta_a \tan \theta_i} \right) & \text{if } (\tan \theta_a \tan \theta_i) > 1 \\ 0 & \text{otherwise} \end{cases} \quad (24)$$

The critical angle ϕ_c^v is determined using the same expression by replacing θ_i with θ_r . The critical angles are related to the masking/shadowing ranges as shown in Table 2.

Using the above critical angle expressions, Table 2, and Table 1, we decompose the integral of equation 23 into the sum of several integrals. Each integral can be evaluated for any given viewer directions. However, for arbitrary directions several cases arise and the results are not easy to use in practice. Therefore, we have chosen to express the radiance of the surface for any arbitrary viewing direction (θ_r, ϕ_r) as a weighted sum of the radiance $L_{rp||}^1$ in the plane of incidence $(\phi_r = \phi_i, \phi_i + \pi)$ and the radiance $L_{rp\perp}^1$ in the perpendicular plane $(\phi_r = \phi_i \pm \frac{\pi}{2})$. We use the following notation: $\alpha = \text{Max}[\theta_i, \theta_r]$ and $\beta = \text{Min}[\theta_i, \theta_r]$; if $\alpha = \theta_i$, $\phi_c^\alpha = \phi_c^s$, else $\phi_c^\alpha = \phi_c^v$; and the same rules apply to ϕ_c^β . The projected radiance in the plane of incidence is:

$$L_{rp||}^1(\theta_a) = \frac{\rho}{\pi} E_0 \cos \theta_i \cos \theta_a \left[1 + \cos(\phi_r - \phi_i) \left(A_1(\theta_a; \alpha) \tan \beta + A_2(\theta_a; \beta, \phi_r - \phi_i) \right) \right] \quad (25)$$

where :

$$A_1(\theta_a; \alpha) = \tan \theta_a \frac{2 \sin \phi_c^\alpha}{\pi} +$$

³ Imagine a V-cavity rotated about the global surface normal for any given source and sensor direction. Various masking/shadowing conditions can be observed.

$$A_2(\theta_a; \beta, \phi_r - \phi_i) = \begin{cases} \frac{1}{2} \tan^2 \theta_a \tan \alpha \left(1 - \frac{2 \phi_c^\alpha + \sin(2 \phi_c^\alpha)}{\pi} \right) \\ \frac{2 \phi_c^\beta}{\pi} - \tan \theta_a \frac{2 \sin \phi_c^\beta}{\pi} \tan \beta & \text{if } \cos(\phi_r - \phi_i) < 0 \\ 0 & \text{if } \cos(\phi_r - \phi_i) \geq 0 \end{cases}$$

Similarly, the projected radiance in the perpendicular plane is determined as:

$$L_{rp\perp}^1(\theta_a) = \frac{\rho}{\pi} E_0 \cos \theta_i \cos \theta_a \left[1 + A_3(\theta_a; \theta_r, \theta_i) \right] \quad (26)$$

$$A_3(\theta_a, \theta_r, \theta_i) = \begin{cases} 0 & \text{if } \phi_c^s + \phi_c^v \leq \frac{\pi}{2} \\ \frac{\frac{1}{2} - \frac{\phi_c^s + \phi_c^v}{\pi}}{\sqrt{\tan^2 \theta_r \tan^2 \theta_a - 1}} + \frac{\frac{1}{2} - \frac{\phi_c^s + \phi_c^v}{\pi}}{\sqrt{\tan^2 \theta_i \tan^2 \theta_a - 1}} - \frac{\tan \theta_a \sqrt{\tan^2 \theta_i + \tan^2 \theta_r}}{\pi} & \text{if } \phi_c^s + \phi_c^v > \frac{\pi}{2} \end{cases}$$

The radiance of the surface in any arbitrary direction is approximated as the following weighted sum of $L_{rp||}^1(\theta_a)$ and $L_{rp\perp}^1(\theta_a)$:

$$L_{rp}(\theta_a) \approx |\cos(\phi_r - \phi_i)| L_{rp||}(\theta_a) + (1 - |\cos(\phi_r - \phi_i)|) L_{rp\perp}(\theta_a) \quad (27)$$

This approximation was obtained by studying the expressions for the radiance components in the two planes. It is very accurate in general, with a slight over-estimation only for $\theta_r = \theta_i$ and $\theta_i \rightarrow \pi/2$. Using the above linear combination of the project radiance in the two planes, we obtain the final expression for the projected radiance:

$$L_{rp}^1(\theta_a) = \frac{\rho}{\pi} E_0 \cos \theta_i \cos \theta_a \left[1 + \cos(\phi_r - \phi_i) \left(A_1(\theta_a; \alpha) \tan \beta + A_2(\theta_a; \beta, \phi_r - \phi_i) \right) + (1 - |\cos(\phi_r - \phi_i)|) A_3(\theta_a; \theta_r, \theta_i) \right] \quad (28)$$

Note that the above projected radiance is the same as the total radiance of the surface $L_r^1(\theta_a) = L_{rp}^1(\theta_a)$ since we have only type of facet slope on the surface ($P(\theta_a) = 1$). In the above derivation, we have not considered multiple reflections as the interreflection component of equation 21 is difficult to intergrate over all cavity orientation angles ϕ_a . However, in the next section we present an approximation to the inter-reflection component for surfaces with Gaussian slope-area distribution.

Partial Shadow	No Shadow	Complete Self-Shadow
$ \phi_a - \phi_i < \phi_c^s$	$\phi_c^s \leq \phi_a - \phi_i \leq \pi - \phi_c^s$	$ \phi_a - (\phi_i + \pi) < \phi_c^s$
Partial Masking	No Masking	Complete Self-Masking
$ \phi_a - \phi_r < \phi_c^v$	$\phi_c^v \leq \phi_a - \phi_r \leq \pi - \phi_c^v$	$ \phi_a - (\phi_r + \pi) < \phi_c^v$

Table 2: Masking/shadowing and the critical angles.

Once again, it is important to note that the radiance of the rough surface considered here is not constant with respect to the viewing direction (θ_r, ϕ_r) . In other words, it behaves in a non-Lambertian manner. We will study this behavior more closely in the following section.

4.3 Model for Gaussian Slope Distribution

The surface considered above consists of V-cavities with a single facet slope. Real surfaces can be modeled only if the slope-area distribution $P(\theta_a, \phi_a)$ includes cavities with several different facet slopes. If the surface roughness is isotropic, the slope-area distribution can be described using a single parameter namely θ_a since the parameter ϕ_a for facet orientation in the surface plane is uniformly distributed. The radiance of any isotropic surface can therefore be determined as:

$$L_r = \int_0^{\frac{\pi}{2}} P(\theta_a) L_{rp}(\theta_a) d\theta_a \quad (29)$$

where the source illumination (no interreflections) component of $L_{rp}(\theta_a)$ is given by equation 28. Here, we assume the isotropic distribution of facet slope to be Gaussian with mean μ and standard deviation σ , i.e. $P(\theta_a; \sigma, \mu)$. Reasonably rough surfaces can be described using a zero mean ($\mu = 0$) Gaussian distribution. The roughness of the surface is then determined by the parameter σ .

The reflectance model can be obtained by using the projected radiance $L_{rp}^1(\theta_a)$ given by equation 28 and the Gaussian distribution $P(\theta_a; \sigma, 0)$ in the integral of equation 29. The resulting integral cannot be evaluated. Hence, we pursued a functional approximation to the integral that is accurate for arbitrary surface roughness and angles of incidence and reflection. In developing this approximation, we carefully studied the functional form of $L_{rp}^1(\theta_a)$. This enabled us to identify some basis functions that can be used in the approximation. Then, we conducted a

large set of numerical evaluations of the integral in equation 29 by varying the surface roughness σ , the angles of incidence (θ_i, ϕ_i) , and the reflection angles (θ_r, ϕ_r) . These simulations and the identified basis functions were used to arrive at an accurate functional approximation for surface radiance. This procedure was applied independently to the source illumination component as well as the interreflection component.

The final approximation results are given below. As before, we define $\alpha = \text{Max}[\theta_r, \theta_i]$ and $\beta = \text{Min}[\theta_r, \theta_i]$. The source illumination component of the radiance of a surface with roughness σ is:

$$L_r^1(\theta_i, \phi_i; \theta_r, \phi_r; \sigma) = \frac{\rho}{\pi} E_0 \cos \theta_i \left[C_1(\sigma, \mu) + \cos(\phi_r - \phi_i) \left(C_2(\alpha, \sigma, \mu) - C_3(\beta; \phi_r - \phi_i; \sigma, \mu) \right) \tan \beta + \left(1 - |\cos(\phi_r - \phi_i)| \right) C_4(\alpha; \beta; \sigma, \mu) \tan \left(\frac{\alpha + \beta}{2} \right) \right] \quad (30)$$

where the coefficients are:

$$\begin{aligned} C_1 &\approx 1 + \left(\frac{2}{\pi} - 1 \right) \frac{\sigma^2}{\sigma^2 + 0.4} \\ C_2 &\approx 0.4 \frac{\sigma^2}{\sigma^2 + 0.21} (\sin \alpha)^{(0.56 + \frac{2.2\sigma}{\pi})} \\ C_3 &\approx \begin{cases} 0.4 \frac{\sigma^2}{\sigma^2 + 0.21} \left(\frac{2\beta}{\pi} \right)^{(2 + \frac{1}{2\sigma^2})} & \text{if } \cos(\phi_r - \phi_i) < 0 \\ 0 & \text{otherwise} \end{cases} \\ C_4 &\approx 0.11 \left(\frac{\sigma^2}{\sigma^2 + 0.21} \right) \left(\frac{4\alpha\beta}{\pi^2} \right)^{(1.55 + \frac{1}{1.55\sigma^2})} \end{aligned}$$

The functional approximation to the interreflection component of surface radiance is:

$$L_r^2(\theta_i, \phi_i; \theta_r, \phi_r; \sigma) = \rho^2 E_0 \cos \theta_i \frac{\sigma^2}{\sigma^2 + 0.4} \left[0.04 - 0.11 \cos(\phi_r - \phi_i) \left(\cos \frac{\alpha}{2} \right)^{3.2} (\sin \beta)^{1.5} \right] \quad (31)$$

The two components are combined using the interreflection parameter χ to obtain the total surface radiance:

$$L_r(\theta_i, \phi_i; \theta_r, \phi_r; \sigma) = L_r^1(\theta_i, \phi_i; \theta_r, \phi_r; \sigma) + \chi L_r^2(\theta_i, \phi_i; \theta_r, \phi_r; \sigma) \quad (32)$$

Finally, the bi-directional reflectance function of the surface is obtained from its radiance and irradiance as $f_r(\theta_i, \phi_i; \theta_r, \phi_r; \sigma) = L_r(\theta_i, \phi_i; \theta_r, \phi_r; \sigma) / E(\theta_i, \phi_i)$. It is important to note that the approximation presented here obeys Helmholtz's reciprocity principle.

In the next section, we present several experimental results that verify the diffuse reflectance model presented here. We conclude this section with a brief illustration of the main characteristics predicted by the model. Figure 6 shows the reflectance predicted by the model for a rough Lambertian surface with roughness $\sigma = 70^\circ$, albedo $\rho = 0.95$, and $\chi = 1$. The radiance L_r in the plane of incidence ($\phi_r = \phi_i, \phi_i + \pi$) is plotted as a function of the reflection angle θ_r for the angle of incidence $\theta_i = 75^\circ$. Two curves are shown in the figure; one generated by numerical integration of equation 29 and the second (plotted as a thin line) obtained by the model approximation given by equations 30 and 31. Notice that these radiance plots deviate substantially from Lambertian reflectance. *The surface radiance increases as the viewing direction approaches the source direction.* The plot can be divided into three sections. In the *backward* (source) direction, the radiance is maximum and gets "cut-off" due to strong masking effects when θ_r exceeds θ_i . This cut-off occurs exactly at $\theta_r = \theta_i$ and is independent of roughness. In the middle section of the plot, radiance varies as a scaled $\tan \theta_r$ function with constant offset. Finally, the interreflections dominate in the *forward* direction where most facets are self-shadowed and the visible facets receive light mainly from adjacent facets. The deviation from pure Lambertian behavior increases with the angle of incidence θ_i .

5 Experiments

We have conducted several experiments to verify the accuracy of the reflectance model presented here. The experimental set-up used to measure the radiance of samples is shown in Figure 7. In the case of outdoor scenes, each sensor element (pixel) typically includes a large surface area (several inches in dimensions and often more). Commercially available reflectance measurement devices are applicable only to small samples. Hence, we developed our own measurement device. Each sample is approximat-

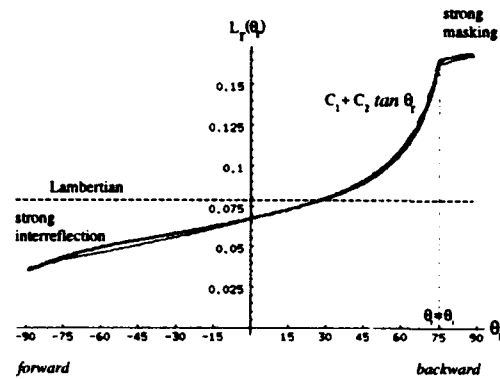


Figure 6: Diffuse reflectance from a surface with roughness $\sigma = 70^\circ$ for angle of incidence $\theta_i = 75^\circ$. The numerical integration is the thick line and the functional approximation is the thin line.

edly 2×2 inches. It is imaged using a 512×480 pixel CCD camera that is mounted at the end of a 6 foot long beam. The other end of the beam is attached to a rotary stage to facilitate precise variation of the viewing angle θ_r . The sample is illuminated using a 300 Watt incandescent light source. The illumination direction (θ_i, ϕ_i) is varied manually. Images of the sample are digitized and the radiance is computed as the average brightness over all pixels that lie on the sample.

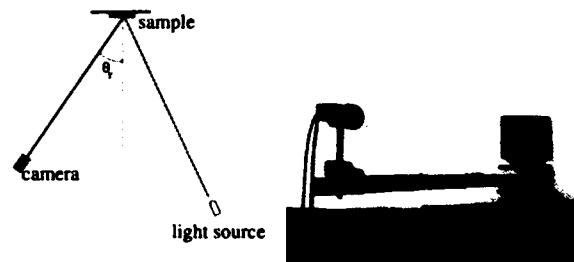


Figure 7: Sketch and photograph of the set-up used to measure reflectance.

Figure 8 shows results obtained for a sample of wall plaster. The sample has matte local reflectance properties but is very rough; it is exactly the type of surface that our diffuse reflectance model characterizes. The reflectance is represented by the radiance of the surface in the sensor direction. The radiance of each sample is plotted as a function of the sensor direction θ_r for different angles of incidence θ_i . These measurements are made in the plane of incidence ($\phi_r = \phi_i = 0$). The dots represent measured radiance values while the solid lines are predictions

obtained using the reflectance model for Gaussian surface roughness. The roughness σ was empirically selected to obtain the best match between measured and predicted radiance.

Similar results are presented in Figures 9 and 10 for sample B (painted sand paper) and sample C (white sand). For three samples, the radiance increases as the viewing direction θ_r approaches the source direction θ_i (backward reflection). This is in contrast to the behavior of rough specular surfaces that reflect more in the forward direction, or Lambertian surfaces where the radiance does not vary with viewing direction. For all three samples, the model predictions and experimental measurements match remarkably well. In all cases, a small peak is noticed near the source direction. This is a different phenomenon from the one described by our model; it is the backscatter peak noticed by others [Oetking, 1966, Hapke and Horn, 1963]. In the case of sample C (sand), we see a small specular component in the forward direction. This due to the specular characteristics of individual sand particles.

6 Implication on Machine Vision

We conclude this paper with a brief discussion on the implication of the diffuse reflectance model presented here on machine vision algorithms. Techniques such as shape from shading and photometric stereo make assumptions regarding the reflectance properties of surfaces in the scene. Incorrect modeling of the reflectance properties naturally leads to inaccurate shape recovery. The reflectance model presented here clearly indicates that rough diffuse surfaces cannot be assumed to be Lambertian in reflectance; they appear brighter in the backward (source) direction rather than the surface normal direction or the forward (specular) direction. Further, this deviation from Lambertian behavior increases with the roughness of the surface and the angle of incident light. The model can therefore be used to improve the performance of shape recovery methods. It can also be used to recover the macroscopic roughness (σ) of surfaces from diffuse reflectance measurements. The recovered roughness can be used to predict the appearance of the surface under different illumination and viewer directions.

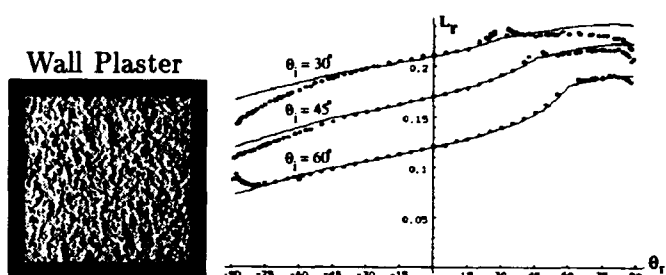


Figure 8: Reflectance measurement and reflectance model (using $\sigma = 50^\circ$, $\chi = 0.75$) plots for wall plaster (sample A). Radiance is plotted as a function of sensor direction (θ_r) for different angles of incidence ($\theta_i = 30^\circ, 45^\circ, 60^\circ$).

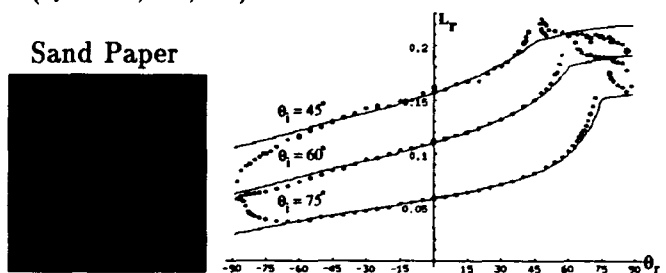


Figure 9: Reflectance measurement and reflectance model (using $\sigma = 70^\circ$, $\chi = 0.50$) plots for painted sand-paper (sample B).

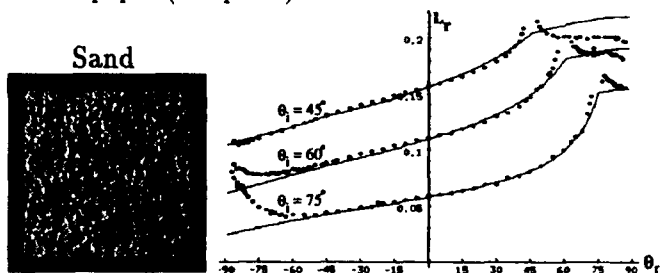


Figure 10: Reflectance measurement and reflectance model (using $\sigma = 70^\circ$, $\chi = 0.50$) plots for sand (sample C).

Figure 11 compares the reflectance map of a Lambertian surface with that of a rough Lambertian surface with $\sigma = 70^\circ$. It is interesting to note that the rough Lambertian surface produces a reflectance map that appears very similar to the linear reflectance map hypothesized for the lunar surface (our model predicts that this linearity of the reflectance map exists only when viewer direction is close to source direction). In a sense, the reflectance model presented here establishes a continuum from pure Lambertian to lunar reflectance by simply varying the roughness of the surface. Finally, the surface roughness model used to derive the reflectance model is the same as the one used by the popular

Torrance and Sparrow [Torrance and Sparrow, 1967] model for specular reflection from rough surfaces. Hence, reflectance from general rough surfaces may be modeled as a linear combination of the Torrance-Sparrow model and the model proposed in this paper.

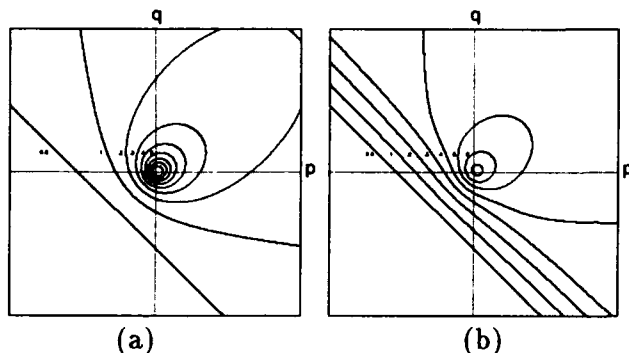


Figure 11: Reflectance maps for (a) Lambertian surface $\rho = 0.9$ (b) rough Lambertian surface ($\sigma = 70^\circ$, $\rho = 0.90$, $\chi = 0.75$). For both maps the angles of incidence are $\theta_i = 10^\circ$ and $\phi_i = 45^\circ$. Note the similarity between the second map and the well-known linear reflectance map previously suggested for lunar reflectance.

References

- [Beckmann, 1965] P. Beckmann. Shadowing of random rough surfaces. *IEEE Transactions on Antennas and Propagation*, AP-13:384-388, 1965.
- [Blinn, 1977] J. F. Blinn. Models of light reflection for computer synthesized pictures. *ACM Computer Graphics (SIGGRAPH 77)*, 19(10):542-547, 1977.
- [Buhl et al., 1968] D. Buhl, W. J. Welch, and D. G. Rea. Reradiation and thermal emission from illuminated craters on the lunar surface. *Journal of Geophysical Research*, 73(16):5281-5295, August 1968.
- [Forsyth and Zisserman, 1989] D. Forsyth and A. Zisserman. Mutual illumination. *Proc. Conf. Computer Vision and Pattern Recognition*, pages 466-473, 1989.
- [Hapke and Horn, 1963] B. W. Hapke and H. Van Horn. Photometric studies of complex surfaces, with applications to the moon. *Journal of Geophysical Research*, 68(15):4545-4570, August 1963.
- [Koenderink and van Doorn, 1983] J. J. Koenderink and A. J. van Doorn. Geometrical modes as a general method to treat diffuse interreflections in radiometry. *Journal of the Optical Society of America*, 73(6):843-850, 1983.
- [Öpik, 1924] E. Öpik. Photometric measures of the moon and the moon the earth-shine. *Publications de L'Observatoire Astronomical de L'Universite de Tartu*, 26(1):1-68, 1924.
- [Minnaert, 1941] M. Minnaert. The reciprocity principle in lunar photometry. *Astrophysical Journal*, 93:403-410, 1941.
- [Nicodemus et al., 1977] F. E. Nicodemus, J. C. Richmond, and J. J. Hsia. *Geometrical Considerations and Nomenclature for Reflectance*. National Bureau of Standards, October 1977. Monograph No. 160.
- [Oetking, 1966] P. Oetking. Photometric studies of diffusely reflecting surfaces with application to the brightness of the moon. *Journal of Geophysical Research*, 71(10):2505-2513, May 1966.
- [Shibata et al., 1981] T. Shibata, W. Frei, and M. Sutton. Digital correction of solar illumination and viewing angle artifacts in remotely sensed images. *Machine Processing of Remotely Sensed Data Symposium*, pages 169-177, 1981.
- [Smith, 1967] B. G. Smith. Lunar surface roughness: Shadowing and thermal emission. *Journal of Geophysical Research*, 72(16):4059-4067, August 1967.
- [Tagare and deFigueiredo, 1991] H. D. Tagare and R. J. P. deFigueiredo. A theory of photometric stereo for a class of diffuse non-Lambertian surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(2):133-152, February 1991.
- [Torrance and Sparrow, 1967] K. Torrance and E. Sparrow. Theory for off-specular reflection from rough surfaces. *Journal of the Optical Society of America*, 57:1105-1114, September 1967.
- [Wagner, 1966] R. J. Wagner. Shadowing of randomly rough surfaces. *Journal of the Acoustical Society of America*, 41(1):138-147, June 1966.

Separation of Reflection Components Using Color and Polarization

Shree K. Nayar, Xi-Sheng Fang, and Terrance Boulton *

Center for Research in Intelligent Systems, Department of Computer Science,
Columbia University New York, N.Y. 10027, U.S.A.

Abstract

Specular reflections and interreflections produce strong highlights in brightness images. These highlights can cause vision algorithms, such as, segmentation, shape from shading, binocular stereo, and motion detection to produce erroneous results. We present an algorithm for separating the specular and diffuse components of reflection from images. The method uses color and polarization, simultaneously, to obtain strong constraints on the reflection components at each image point. Polarization is used to locally determine the color of the specular component, constraining the diffuse color at a pixel to a one dimensional linear subspace. This subspace is used to determine neighboring pixels whose color is consistent with a given pixel. Diffuse color information from consistent neighbors is used to determine the diffuse color of the pixel. In contrast to previous separation algorithms, the proposed method can handle highlights that have a varying diffuse component as well as highlights that include regions with different reflectance and material properties. We present several experimental results obtained by applying the algorithm to complex scenes with textured objects and strong interreflections.

1 Introduction

Reflection of light from surfaces can be classified into two broad categories: diffuse and specular. The *diffuse component* results from light rays penetrating the surface, undergoing multiple reflections and refractions, and re-emerging

at the surface. This component is distributed in a wide range of directions around the surface normal, giving the surface a matte appearance. If the viewing direction of an image sensor is varied, diffuse reflections from scene points change slowly and in the ideal case of Lambertian surfaces, it does not change at all. The *specular component*, on the other hand, is a surface phenomenon. Light rays incident on the surface are reflected such that the angle of reflection equals the angle of incidence. Even for marginally rough surfaces, the specular reflections are concentrated in a compact lobe around the specular direction. This concentration of light energy causes strong *highlights* in brightness images of scenes. These highlights can cause vision algorithms for scene segmentation and shading analysis to produce erroneous results. If the sensor direction is varied, highlights shift, diminish rapidly, or suddenly appear in other parts of the scene. This strong directional dependence of specular reflection, poses serious problems for vision techniques such as binocular stereo and motion detection. Hence, specularities are often undesirable in images.

In this paper, we present an algorithm that separates the diffuse and specular components of brightness from images. Separation of reflection components has been a topic of active research in the past few years. We discuss only those efforts that have resulted in algorithms that have been tested on real images. Most of this work is based on the dichromatic reflectance model proposed in [Shafer, 85]. The dichromatic model suggests that, in the case of dielectrics (non-conductors), the diffuse component and the specular component generally have different spectral distribu-

*This work supported in part by DARPA contract DACA-76-92-C-007 and NSF contract #IRI-90-57951

tions. Hence, the color of an image point can be viewed as the sum of two vectors with different directions in color space. Using this model, [Klinker, 88] and [Gershon, 87] independently observed that color histogram of an object with uniform diffuse color takes the shape of a skewed T with two limbs. One limb corresponds to purely diffuse points on the object, which have the same color but differ in magnitude, and the second limb represents a highlight region. They proposed algorithms for automatically identifying the two limbs and used the directions of the limbs to separate the diffuse and specular components at each object point. Later, [Bajcsy, et al., 90] showed that the color histogram of an object could have additional limbs that correspond to highlights caused by interreflections between objects. More recently, [Lee, 91] proposed moving a sensor and applying spectral differencing to color histograms of consecutive images to identify specular points in the image. This method however does not compute accurate estimates of the specular component at each image point.

All of the above algorithms rely solely on color information to separate specular and diffuse reflections from images. Since the separation is not possible when an image point is treated in isolation, these methods analyze the anatomy of color histograms. Two major limitations result from the above approach. First, real scenes are complex and include objects with texture and varying reflectance. Color histograms of such scenes are generally unpredictable and a set of linear clusters such as the skewed T are unlikely. Second, all points on the highlight region are assumed to have the same diffuse component (color and magnitude). Even for an object with uniform reflectance, this assumption is valid only if the object surface is very smooth. In the case of rough surfaces, the highlights spread over a wider range of surface normals and the specular limb of the skewed T does not have a well-defined direction.

Recently, [Wolff and Boult, 91] proposed a polarization based method for separating specular and diffuse components from gray-level (black and white) images. Details of this method will be discussed later. Their polarization-based method assumes that the diffuse component (color and magnitude) is constant over the en-

tire highlight region. They also assume that the material type and surface normal do not vary within the highlight region. Using these assumptions, an estimate for a constant diffuse term is obtained. We will show later that the assumptions made in [Wolff and Boult, 91] are often not practical in the context of real scenes.

This paper presents a new algorithm for the separation of specular and diffuse reflection components from images. This algorithm uses color and polarization *simultaneously*, to obtain new constraints of the reflection components. As a result, it does not suffer from many of the problems associated with previous methods based either color or polarization. We assume that the scene consists of dielectric objects. This leads to two assumptions: (a) the dichromatic model is applicable, and (b) the specular component is polarized while the diffuse component is not. The restrictions imposed by these assumptions are discussed in subsequent sections. The proposed algorithm can estimate specular components that result not only from direct source illumination but also interreflections between points in the image. We show that, under reasonable assumptions, polarization information can be used to obtain the color of the specular component independently for each point with a specular component. For each such point the result is a line in color space on which the diffuse vector must lie. This line imposes strong constraints on the color of the diffuse component of that image point. Neighboring diffuse colors that satisfy these constraints are used to compute the diffuse component of the image point.

Since the specular color of each image point is computed independently, our approach has the following advantages over previous methods: (a) The diffuse component is not assumed to be constant under the highlight region; (b) The Fresnel ratio (which depends on the material properties and the angle of incidence) can vary over highlight regions; and (c) the specular component may be textured. The algorithm requires that each image point has a few (at least three) neighbors that have the same computed diffuse color (that is, direction in color space but not necessarily magnitude). Color Figure 1a (see pages with color photos) shows a highlight that spreads over an object with strong surface tex-

ture. Color Figure 1b shows the diffuse image of the object recovered using the proposed algorithm. In the experimental section, we present several other results obtained by applying the algorithm to complex scenes with multiple high-lights and interreflections.

2 Reflection and Interreflection

We begin by describing the mechanisms involved in the processes of reflection and interreflection. Specular reflections cause highlights in the image that pose serious problems for a variety of vision techniques. The objective of this paper is to remove specularities due to single reflections as well as interreflections.

Figure 1 shows two points, *A* and *B*, in a scene. Reflection from the point *A* has two components, namely, diffuse and specular.¹ The diffuse component arises from the scattering of light rays that enter the surface and undergo multiple reflections and refractions. The specular component, on the other hand, is a surface phenomenon and results from single reflection of incident light rays. The surface may be assumed to be composed of several planar elements, or facets, where each facets has its own orientation. The result is a specular component that spreads around the specular direction, the width of the distribution depending on the roughness of the surface [Torrance and Sparrow, 67].

Now let us consider the phenomenon of *inter-reflections*. Points in the scene receive light not only from the light sources but also from other scene points. Assume that point *B* reflects, into the sensor, light energy from the direction of point *A*. The resulting image brightness value can be viewed as the linear combination of four possible interreflection components: (a) *diffuse-diffuse*; (b) *specular-diffuse*; (c) *diffuse-specular*; and (d) *specular-specular*. In each case, the first term represents the component received from point *A* and the second represents the component reflected by point *B*. In general, point *B* could reflect light due to both direct illumination by light sources as well as interreflections from other scene points. We consider each brightness

¹Recently, [Nayar, et. al., 91] proposed a reflectance framework that includes three primary components of reflection: the *diffuse lobe*; the *specular lobe*; and the *specular spike*. In this paper, the two specular components can be combined to yield, the *specular component*.

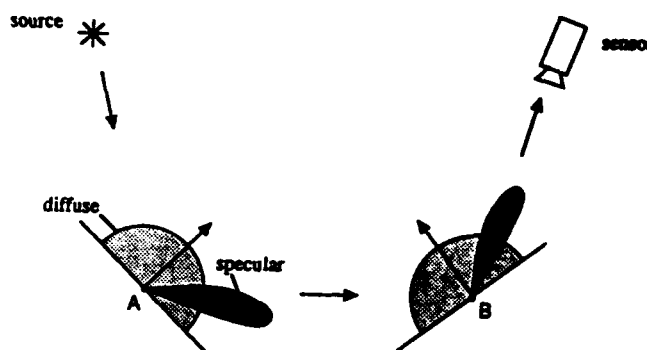


Figure 1: Components of reflection and inter-reflection.

value in the image as the sum of two components, diffuse and specular. The specular component can result either from direct illumination by a light source or due to the diffuse-specular or specular-specular interreflections. We assume that the specular reflection received by the sensor from any given point is either due to source illumination or due to interreflections and not both. In other words, any given scene point is positioned and oriented to specularly reflect essentially from either a light source or another scene point but not both. This assumption holds well except for very rough surfaces.

3 Polarization

The method presented in this paper uses a polarization filter to determine the color of the specular component. In this section, we present a brief overview of polarization and discuss the type of surfaces for which it provides useful information. Detailed discussions on the theory of polarization can be found in [Born and Wolf, 65]. In the field of machine vision, polarization methods were first introduced in [Koshikawa, 79] who used ellipsometry for shape interpretation and recognition of glossy objects. More recently, [Wolff and Boulton, 91] examined the use of linear polarization for highlight removal and material classification.

Figure 2 shows a surface element illuminated by a source and imaged by a sensor. A polarization filter is placed in front of the sensor. As in the previous section, let the image brightness value corresponding to the surface element be

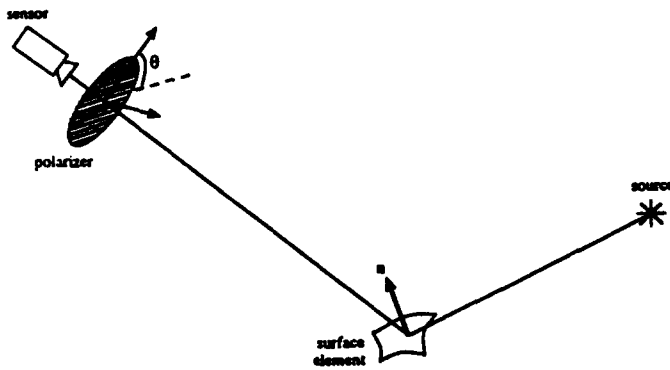


Figure 2: Surface element illuminated by a source and imaged through a polarization filter.

as: $I = I_d + I_s$, where, I_d is the diffuse component and I_s is the specular component. The linear polarization of each light wave is determined by the direction of its electric field vector. In general, the light energy (due to several light waves) reflected by a surface may be partially *polarized*. The extent of polarization depends on several factors including the material of the reflecting surface element, its orientation with respect to the image sensor, and the types of reflection mechanisms (specular or diffuse) at work.

The diffuse component of reflection tends to be *unpolarized*.² In contrast, the specular component tends to be partially polarized; rotation of the polarization filter varies the specular component as a cosine function, as shown in Figure 3. The specular component can be expressed as the sum of a *specular constant* I_{sc} and a *specular varying* term that is a cosine function with amplitude I_{sv} :

$$I = I_d + I_{sc} + I_{sv} \cos 2(\theta - \alpha) \quad (1)$$

where, θ represents the angle of the polarization filter and α is the phase angle determined by the projection of the normal of the surface element onto the plane of the polarization filter. The exact values of I_{sc} and I_{sv} depend on the material properties and the angle of incidence. This dependence is determined by the *Fresnel reflection coefficients* $F_{\perp}(\eta, \psi)$ and $F_{\parallel}(\eta, \psi)$ which represent the polarization of the reflected light

²Note that this assumption does not hold near the occluding contour of an object, see [Boult and Wolff, 91]. That paper addresses the classification of scene edges based on their polarization characteristics.

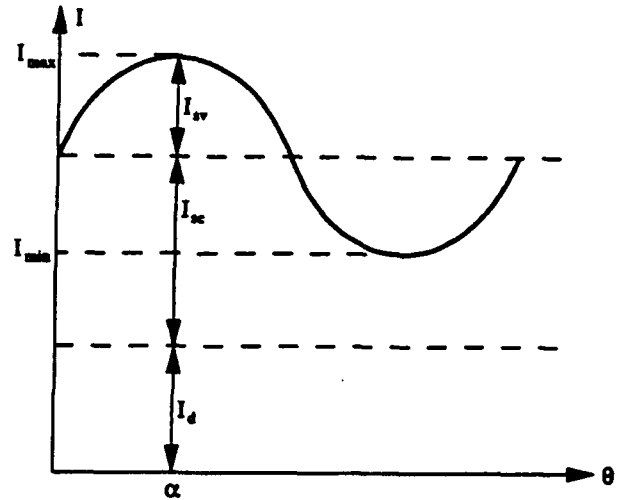


Figure 3: Image brightness plotted as a function of polarization filter position.

waves in the directions perpendicular and parallel to the plane of incidence, respectively. The relationship between I_{sc} and I_{sv} and the Fresnel coefficients is:

$$\frac{I_{sc} + I_{sv}}{I_{sc}} = \frac{F_{\perp}(\eta, \psi)}{F_{\parallel}(\eta, \psi)} \quad (2)$$

The parameter η is the *complex index of refraction* of the surface medium, and depends on the properties of the reflecting material. The parameter ψ is the angle of incidence.³

Note that the terms I_d and I_{sc} in equation 1 are constant and can be represented by a single component $I_c = I_d + I_{sc}$ to obtain: $I = I_c + I_{sv} \cos 2(\theta - \alpha)$. For any given position θ_i of the polarization filter we have:

$$I_i = I_c + I_{sv} \cos 2(\theta_i - \alpha). \quad (3)$$

This can also be represented as the dot product of two vectors:

$$\begin{aligned} \mathbf{f}_i &= (1, \cos 2\theta_i, \sin 2\theta_i) \\ \mathbf{v} &= (I_c, I_{sv} \cos 2\alpha, I_{sv} \sin 2\alpha) \\ I_i &= \mathbf{f}_i \cdot \mathbf{v}. \end{aligned} \quad (4)$$

Let M be the total number of discrete filter positions used to obtain the image brightness values

³For metals, the two Fresnel coefficients are nearly equal except close to the grazing angle (when ψ lies between 70 and 90 degrees). Thus, linear polarization based methods are generally not effective for metals. For dielectrics (non-conductors), however, the two Fresnel coefficients differ substantially except for near-normal angles of incidence (when ψ is less than 10-15 degrees).

$\{I_i \mid i = 1, 2, \dots, M\}$. If $M = 3$, equation 4 yields a linear system of equations that can be solved to obtain the parameters I_c , I_{sv} , and α . If $M > 3$, we have an over-determined linear system that can be solved to obtain more robust estimates of I_c , I_{sv} , and α , in the presence of image noise.⁴

From I_c and I_{sv} , we can obtain the maximum and minimum values of image brightness as:

$$I_{\min} = I_c - I_{sv}, \quad I_{\max} = I_c + I_{sv} \quad (5)$$

The *degree of polarization* at a scene point can be determined as [Born and Wolf, 65]:

$$\rho = \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}} \quad (6)$$

The degree of polarization lies between 0 and 1 and can be used during highlight removal to classify points into those that are only diffuse ($\rho \ll 1$) and those that include a specular component. However, this measure must be used with care as both I_{\min} and I_{\max} include the constant specular component I_{sc} as well as the diffuse component I_d ; varying either I_{sc} or I_d has the same effect on ρ .

We conclude this section with a note on previous work on highlight removal using polarization. A method for computing I_d and I_s by rotating the polarization filter, is presented in [Wolff and Boulton, 91]. From the above discussion, we know that I_d and I_{sc} are both constant. They can be computed from I_c only if we know the ratio of the Fresnel coefficients, $q = F_{\perp}(\eta, \psi)/F_{\parallel}(\eta, \psi)$, for the corresponding scene point. The Fresnel coefficients are determined by the material properties of the scene point as well as the angle of incidence. Neither of these factors are known. To constrain the problem, [Wolff and Boulton, 91] use all points (pixels) on a segmented highlight. They assume that both the diffuse component I_d as well as the Fresnel ratio q are constant under the highlight region, and estimate them using all points within the highlight region. This assumption, however, is unrealistic for two reasons. First, in real scenes, the diffuse component I_d within the highlight region may vary due to

the curvature of the surface or due to texture on the surface. Secondly, the Fresnel ratio q cannot be assumed to be constant since the angle of incidence can vary substantially over large highlight regions resulting from extended sources in the scene. The latter of these problems was discussed in [Wolff, 90] but no solutions were proposed.

4 Color

We now discuss the role of color in the removal of specularities. We present an overview of the representation of diffuse and specular components in color space and discuss previous work on the removal of highlights using color. In contrast to gray-level images, color images represent wavelength (λ) dependence of the light reflected by a scene. Let $x(\lambda)$ be the spectral distribution of the light reflected by a scene point, and $s(\lambda)$ represent the response of the sensor to wavelength.

Typically, color images are obtained by using three filters with responses $r(\lambda)$, $g(\lambda)$, and $b(\lambda)$ that have peaks close to the wavelengths that humans perceive as "red," "green," and "blue." The resulting three brightness values measured by a sensor element constitute the color vector $\mathbf{I} = [I^r, I^g, I^b]$ for the corresponding point in scene. The three brightness values in the color vector are related to the spectral distribution of the reflected light as:

$$\begin{aligned} I^r &= \int x(\lambda) r(\lambda) s(\lambda) d\lambda \\ I^g &= \int x(\lambda) g(\lambda) s(\lambda) d\lambda \\ I^b &= \int x(\lambda) b(\lambda) s(\lambda) d\lambda \end{aligned} \quad (7)$$

Each brightness value includes a diffuse component and a specular component. Hence, in three-dimensional color space we have the following decomposition: $\mathbf{I} = \mathbf{I}_d + \mathbf{I}_s$.

The *dichromatic reflectance model* [Shafer, 85] suggests that, for dielectrics, the spectral distribution of the diffuse component is determined by the colorant in the surface whereas the specular component preserves the spectral distribution of the incident light. As a result, the two vectors \mathbf{I}_d and \mathbf{I}_s generally have different directions in color space. The two vectors will however have

⁴This formulation of the problem using vectors saves substantial computations compared to the non-linear formulation of the type $a + b \sin^2(\theta - \alpha)$ used in [Boulton and Wolff, 91] and [Wolff, 90] that require the use of iterative non-linear estimation techniques.

the same direction if, for instance, a gray object is illuminated by white light.

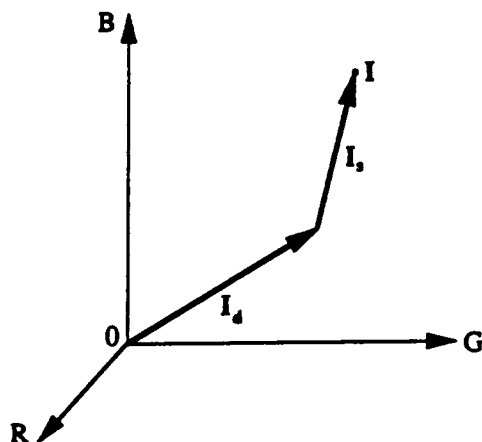


Figure 4: Diffuse and specular components in color space.

As discussed in the introduction, [Klinker, 88] and [Gershon, 87] independently used the dichromatic reflectance model to remove highlights from images by the two limbs of the skewed T in color space.⁵ These methods are based on two main assumptions: (a) the object is segmented away from the scene and has a single uniform diffuse color and (b) the diffuse component is near constant within the highlight region. These two assumptions must hold for a skewed T to be formed in color space. The first assumption is often violated in real scenes where objects may be textured or have patches with different reflectance properties. The second assumption can be valid only if the surface is very smooth, thus producing a very compact highlight. Even for marginally rough surfaces, the highlight is expected to include object points with a range of surface orientations. In such cases, the specular limb of the skewed T spreads into a wide cluster in color space that is difficult to separate from the diffuse limb. This last observation has also been made by Novak and Shafer [Novak and Shafer, 92].

⁵Along the same lines, [Bajcsy, et. al., 90] observed that the color histogram can include additional limbs that correspond to highlights caused by interreflections. The highlight removal algorithms of [Klinker, 88] and [Gershon, 87] could, in theory, be modified to identify additional limbs and remove specularities due to interreflections.

5 Removal of Specularities using Color and Polarization.

In this section, we develop a method for removing specular reflections from images. We make the following assumptions.

- (A) The dichromatic reflectance model applies at each point. Hence the scene consists of dielectric objects, and therefore specular reflections and interreflections are polarized while the diffuse reflections are not. Further, the color of the incident light at each scene point is different from the color of the material.
- (B) The specular interreflections result from either the diffuse-specular mechanism *or* the specular-specular mechanism and not both. In the first case, the incident light is unpolarized and hence the Fresnel ratio is simply $q = F_{\perp}(\eta, \psi) / F_{\parallel}(\eta, \psi)$. In the second case, the incident light is partially polarized and the effective Fresnel ratio for the surface point is $q = aF_{\perp}(\eta, \psi) / bF_{\parallel}(\eta, \psi)$. The parameters a and b account for the partial polarization of the incident light.
- (C) Fresnel coefficients $F_{\perp}(\eta, \psi)$ and $F_{\parallel}(\eta, \psi)$ are independent of the wavelength of incident light. This assumption is reasonable (see [Driscoll, 78]) since we are assuming dielectrics and operating in the visible-light spectrum. Assumptions (B) and (C) result in the Fresnel ratio q being equal for all three color bands.

The color of the specular component is computed *locally* at each pixel. This places strong constraints on the diffuse component I_d . Neighboring diffuse points that satisfy these constraints are then used to compute the diffuse component. This approach has the following advantages over all of the previous methods for highlight removal:

- In contrast to the previous methods based either on color or polarization, we do not assume that the diffuse component I_d is constant within each highlight region. In fact, the surfaces could be textured with patches of different materials underlying the highlights.
- The specular color of each point is computed independently. This *local* approach

does not require prior segmentation of either highlights or objects in the scene. Further, the highlights need not be compact; the method can handle substantial surface roughness conditions.

We describe the technique for specular removal by focusing on a single image point \mathbf{x} . The same procedure is applied independently to all image points. The color vector for the image point is: $\mathbf{I} = \mathbf{I}_d + \mathbf{I}_s$. Given the above assumptions, the Fresnel ratio q is the same for all three color bands. Hence the cosine term in equation 4 will be in phase for the 3 color bands and for the polarization filter position θ_i we have the color vector: $\mathbf{I}_i = \mathbf{I}_c + \mathbf{I}_{sv} \cos 2(\theta_i - \alpha)$. In our experiments, we have used 6 or more polarizer positions. This gives us an over-determined linear system of equations that are solved to obtain robust estimates of \mathbf{I}_c , \mathbf{I}_{sv} , and α . The color vectors corresponding to maximum and minimum polarization (see equation 5) are:

$$\begin{aligned} \mathbf{I}_{\max} &= \mathbf{I}_c + \mathbf{I}_{sv} \\ \mathbf{I}_{\min} &= \mathbf{I}_c - \mathbf{I}_{sv} \end{aligned}$$

Two tests are used to determine if the image point \mathbf{x} is to be processed any further. First, a degree of polarization ρ (expression 6) is computed for each of the three color bands. If the largest of three ρ estimates is less than a threshold value T_1 , the point is not sufficiently polarized and is assumed to be purely diffuse. In this case, the next image point is examined.

If the degree of polarization of the point \mathbf{x} is greater than T_1 , the angle β subtended by the vector $\mathbf{k} = \mathbf{I}_{\max} - \mathbf{I}_{\min}$ from the origin $\mathbf{0}$ is computed (see Figure 5). If β is less than a threshold T_2 , the color of the specular component is very similar to that of the diffuse component \mathbf{I}_d , and the dichromatic model cannot be used with confidence.

On the other hand, if the point \mathbf{x} is polarized and its β value is not small, we proceed to compute its diffuse component \mathbf{I}_d . If we can determine \mathbf{I}_{sc} , then the diffuse component can be computed as $\mathbf{I}_d = \mathbf{I}_c - \mathbf{I}_{sc}$. Once this is done, the specular component \mathbf{I}_s can be separated from any one of the images \mathbf{I}_i . We can also obtain an approximation to the specular image we would see without a polarizer by subtracting \mathbf{I}_d from $\frac{1}{2}(\mathbf{I}_{\max} + \mathbf{I}_{\min})$.

Recall that the specular components \mathbf{I}_{sc} and \mathbf{I}_{sv} satisfy $\mathbf{I}_{sc} + \mathbf{I}_{sv} = \mathbf{I}_{sc} q$. Unfortunately, the Fresnel coefficient q is not known as it depends on the material properties and the angle of incidence. Though we have estimates of \mathbf{I}_c and \mathbf{I}_{sv} , we do not have a simple way of determining \mathbf{I}_d . The color measurements \mathbf{I}_i obtained by rotating the polarization filter lie on a straight line L (see Figure 5) in color space. The diffuse component \mathbf{I}_d is unaffected by rotations of the polarizer; only the specular component \mathbf{I}_s varies. The specular component varies along a straight line since the cosine functions in the three color bands are in phase (assumptions B and C).

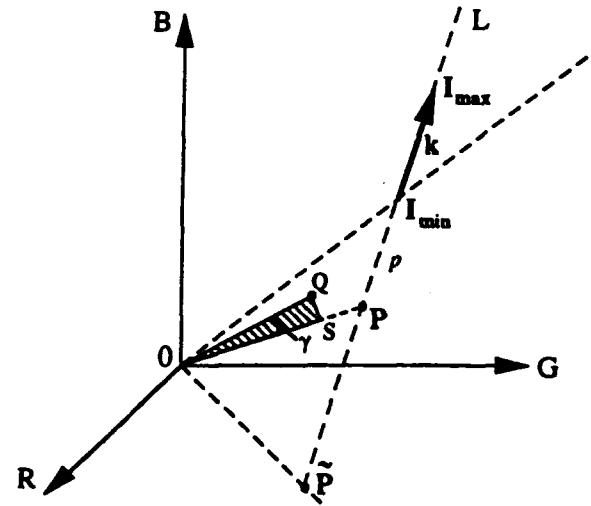


Figure 5: Using neighboring points and specular line constraint to compute the diffuse component \mathbf{I}_d .

Though we are unable to compute the diffuse component locally, the specular line gives useful constraints on the diffuse component. These constraints are applied to the neighboring pixels of \mathbf{x} to obtain the diffuse component \mathbf{I}_d . Assume that the diffuse component of \mathbf{x} corresponds to the point \mathbf{P} . The position of \mathbf{P} on the specular line L can be parametrized as follows: $\mathbf{P} = \mathbf{I}_{\min} - p \mathbf{k}$ where p ($0 \leq p \leq \tilde{p}$) is the distance of \mathbf{P} from \mathbf{I}_{\min} as shown in Figure 5 and \tilde{p} is defined as:

$$\tilde{p} = \text{Min} \left(\frac{I_{\min}^r}{k^r}, \frac{I_{\min}^g}{k^g}, \frac{I_{\min}^b}{k^b} \right). \quad (8)$$

\tilde{p} determines the point $\tilde{\mathbf{P}}$ that is the intersection of the specular line L with one of the three

planes of the color space. In the example shown in Figure 5, the specular line intersects with the $R - G$ plane. In general, however, L could intersect any one of the three planes that constitute the color space. Define \mathcal{P}_1 to be the plane that passes through the points $(0, \mathbf{I}_{\min}, \mathbf{I}_{\max})$. The expression for \mathcal{P}_1 is: $A I^r + B I^g + C I^b = 0$, where:

$$\begin{aligned} A &= k^g I_{\min}^b - k^b I_{\min}^g \\ B &= k^b I_{\min}^r - k^r I_{\min}^b \\ C &= k^r I_{\min}^g - k^g I_{\min}^r. \end{aligned} \quad (9)$$

Since we do not have sufficient constraints to compute the diffuse component \mathbf{I}_d of the point \mathbf{x} from the color measurements \mathbf{I}_i , we use neighboring image points that satisfy the following conditions:

- (1) A neighboring image point \mathbf{y} can be used if we know its diffuse component \mathbf{Q} . This occurs if \mathbf{y} has a low degree of polarization ρ and hence can be assumed to be purely diffuse, or if its diffuse component has already been computed.
- (2) In color space, the vector \mathbf{Q} must lie on the plane \mathcal{P}_1 . Further, it must lie between the vectors \mathbf{I}_{\min} and $\tilde{\mathbf{P}}$, since the diffuse vector \mathbf{I}_d of the point \mathbf{x} lies on the line L between the points \mathbf{I}_{\min} and $\tilde{\mathbf{P}}$. Note, however, that \mathbf{Q} can lie inside or outside the triangle $(0, \mathbf{I}_{\min}, \tilde{\mathbf{P}})$.

If these conditions are satisfied, the neighboring point \mathbf{y} is assumed to have the same diffuse color as the point \mathbf{x} . Then, the line passing through \mathbf{Q} and the specular line L intersect to give \mathbf{P} , an estimate of the diffuse component of \mathbf{x} .

Due to noise in the color and polarization measurements, the diffuse component \mathbf{Q} of a neighboring point is not expected to exactly satisfy the above conditions. To accommodate for such discrepancies, we compute the angle γ subtended by \mathbf{Q} with respect to \mathcal{P}_1 (see Figure 5):

$$\sin \gamma = \frac{A Q^r + B Q^g + C Q^b}{\sqrt{A^2 + B^2 + C^2}} \quad (10)$$

If γ is larger than a threshold value T_3 , \mathbf{Q} is not used any further.⁶ If γ is small ($\gamma \leq T_3$), the

⁶Note that we have used the angle γ rather than the distance of \mathbf{Q} from \mathcal{P}_1 . This is because a neighboring point may

point \mathbf{P} is computed by extending the vector \mathbf{S} to intersect the line L as shown in Figure 5. This can be done without computing the projection \mathbf{S} of \mathbf{Q} on the plane \mathcal{P}_1 . Consider the plane \mathcal{P}_2 that passes through the points $(0, \mathbf{Q}, \mathbf{P})$. It can be expressed as $D I^r + E I^g + F I^b = 0$, where:

$$\begin{aligned} D &= Q^g P^b - Q^b P^g \\ E &= Q^b P^r - Q^r P^b \\ F &= Q^r P^g - Q^g P^r \end{aligned} \quad (11)$$

Since the planes \mathcal{P}_1 and \mathcal{P}_2 are perpendicular, we have: $AD + BE + CF = 0$, which can be expanded using equations 9 and 11. By substituting the expression for \mathbf{P} given by equation 8 in the expansion, a solution for the line parameter p is directly obtained:

$$p = \frac{\begin{pmatrix} A (Q^g I_{\min}^b - Q^b I_{\min}^g) \\ + B (Q^b I_{\min}^r - Q^r I_{\min}^b) \\ + C (Q^r I_{\min}^g - Q^g I_{\min}^r) \end{pmatrix}}{\begin{pmatrix} A (Q^g k^b - Q^b k^g) \\ + B (Q^b k^r - Q^r k^b) \\ + C (Q^r k^g - Q^g k^r) \end{pmatrix}} \quad (12)$$

The above process is repeated for all neighboring diffuse components \mathbf{Q}_j that satisfy conditions (1) and (2). The result is a set of estimates $\{p_j \mid j = 1, 2, \dots, N\}$. If $N < T_4$ (we use $T_4 = 3$ in our implementation) there are not enough neighboring diffuse components to compute a robust estimate of \mathbf{I}_d for the point \mathbf{x} . If $N \geq T_4$, the mean and standard deviation of p_j are computed as:

$$\begin{aligned} \bar{p} &= \frac{\sum_{j=1}^N w_j p_j}{\sum_{j=1}^N w_j} \\ \sigma_p &= \frac{\sum_{j=1}^N (\bar{p} - p_j)^2}{N - 1} \end{aligned} \quad (13)$$

where the weight w_j given to each p_j equals the magnitude of the corresponding diffuse component, $\|\mathbf{Q}_j\|$. The mean value \bar{p} is accepted if the standard deviation σ_p is less than a threshold T_5 , i.e. the estimates p_j form a compact cluster

have a very small diffuse component that lies close to the origin 0 and as a result is also close to plane \mathcal{P}_1 . Such a point could have relatively large errors due to image noise and must not be used in the computation of \mathbf{I}_d .



(a) I_{avg} for cup

(b) I_d (diffuse image)

(c) I_{min}



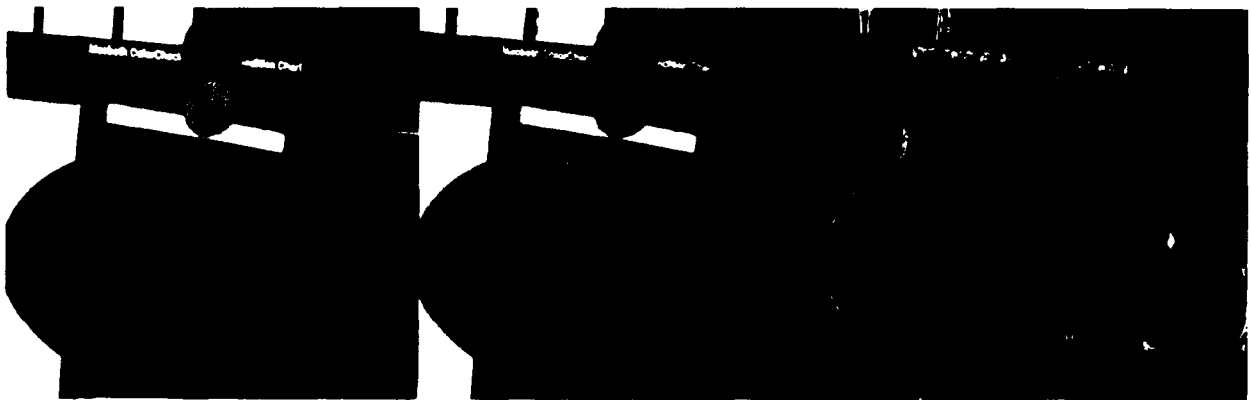
(d) I_{avg}

(e) I_d (diffuse image)

(f) I_s (specular image)

(g) I_{min}

Color Figure 1: Recovery algorithm applied to a highly textured cup. (a) Original (I_{avg}) image. (b) Recovered diffuse image I_d . (c) I_{min} , the image with minimum intensity transmitted through the polarizer. (d) - (g) show results for a 10X10 window centered around the pixel [150,150].

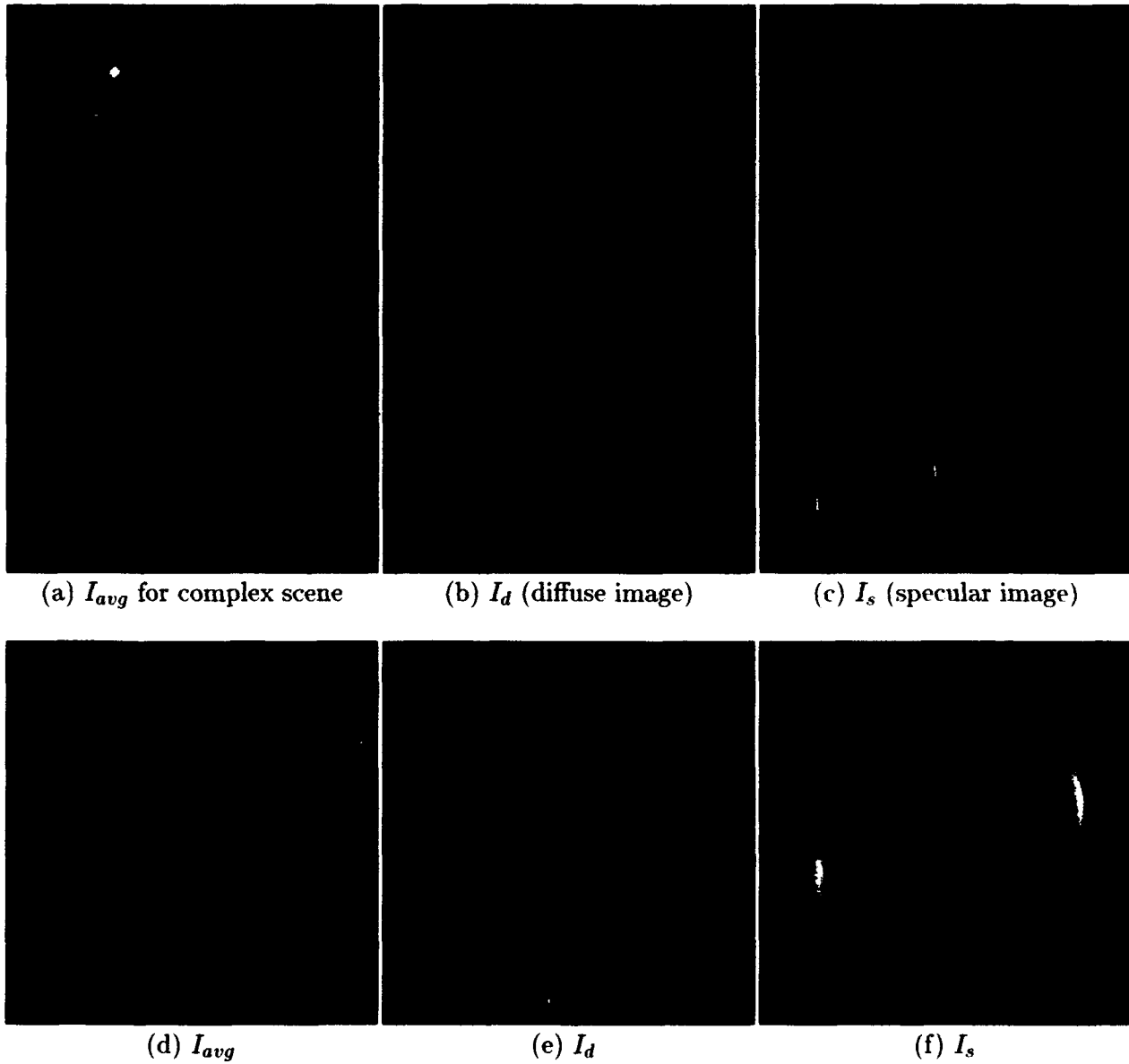


(a) I_{avg} for plate scene

(b) I_d

(c) I_s

Color Figure 2: Recovery algorithm applied to a plastic plate with strong interreflections (and a few strong highlights). (a) Original (I_{avg}) image. (b) Recovered diffuse image I_d . (c) Recovered specular image I_s . (d) Computed percent polarization.



Color Figure 3: Recovery algorithm applied to a very complex scene. (a) Original (I_{avg}) image. (b) Recovered diffuse image I_d . (c) Recovered specular component I_s . (d)-(f) show a close up of the blue torus from the lower left corner of the scene.

on the line L . This constraint is used to ensure that different diffuse colors in the neighborhood of \mathbf{x} that happen to lie close to the plane \mathcal{P}_1 , are not used together to obtain an erroneous estimate of \mathbf{I}_d . Once \bar{p} has been determined, the diffuse component $\mathbf{I}_d = \mathbf{P}$ of the image point \mathbf{x} is obtained using equation 8.

Figure 6 compares the above algorithm with previous techniques based on either polarization or color. The comparison is done using the 40X40 image window shown in Color Figure 1d. The image window includes a highlight that spreads over regions of different diffuse color. Figure 6(a) shows the histogram in (R, G, B) color space for the image window. Note that the anatomy of the histogram is very complex and does not lend itself to the skewed T analysis proposed in [Klinker, 88]. Figure 6(b) shows the cluster obtained in the (I_{\min}, I_{\max}) space for the green band of the image window.⁷ The polarization based method for highlight removal proposed in [Wolff and Boult, 91] is based on the assumption that this cluster in (I_{\min}, I_{\max}) space is linear. As seen in the Figure 6(b), the cluster does not form a straight line and hence the polarization based method is not applicable. Finally, Figure 6(c) shows the result obtained using the algorithm proposed in this paper. The figure shows I_{\min} , I_{\max} , and the specular line computed for the center pixel of the window shown in Color Figure 1(d). The color space constraints are used to select neighboring diffuse colors and compute the diffuse component \mathbf{I}_d of the pixel.

The algorithm proposed in this section is applied to all points in the image. Not all image points end up with an \mathbf{I}_d estimate. An image point may lie in the middle of a very large highlight, in which case, it may not have a sufficient number of neighbors with diffuse colors that satisfy conditions (1) and (2) or produce a compact cluster of intersection points on the line L . Hence, we apply the algorithm repeatedly to the image points. This iterative approach is effective in the case of complex scenes; each iteration provides a new set of computed diffuse colors thus increasing the likelihood of finding neighboring diffuse colors in the next iteration. A large highlight, for instance, shrinks in size with each iteration. The

iterations are discontinued when no new diffuse estimates are obtained.

6 Experimentation

This section presents experimental results obtained using the proposed algorithm. Here we begin with very a short description of the experimental setup and a few details of the implementation of the algorithm. This is followed by results obtained by applying the algorithm to scenes with textured objects, primary (source) specularities, and secondary (interreflection) specularities.

Experimental setup is important as we need both registration between color bands and also linearization of camera response. Details of our setup can be found in [Nayar *et al.*, 1993]. That report also contains more implementational details and experimental examples.

It is worth noting, that the current setups have some minor problems that might be overcome using precision filter mounts and simple correction methods. The first is that the movement of the polarizing filter can cause slight shifts of the image (approximately 1 pixel). The second is that we have not corrected for chromatic aberration of the lens. Finally, we are using CCD technology which is prone to blooming affects near strong highlight regions. All of these problems manifest themselves as errors in polarization fitting (and hence specular removal) in small (1-2 pixel) neighborhoods of scene boundaries.

6.1 Implementation Details

For each scene, a set of color images are obtained by rotating the polarization filter. The images are first corrected using the calibration data. Then, the polarization parameters I_{\min} , I_{\max} , and α are computed for each color channel. These parameters are computed using the linear least squares (LS) fitting method. The results of the polarization fitting are 6 images for each color channel, I_{\min} , I_{\max} , $I_{\text{avg}} (= \frac{I_{\max} + I_{\min}}{2})$, ρ (percent polarization), phase (the angle α), and RMSE (root mean square error in fitting). Of these, only the I_{\max} and I_{\min} images are directly used by the specular removal algorithm. The others are used only to debug the algorithm and analyze the results. The I_{avg} image is what would be obtained without a polarizer but with

⁷The green band was selected as the average degree of polarization for the image window is maximum in this band.

a 50% neutral density filter instead. The RMSE gives the error in fitting the image data to equation 1, and most pixels in the image have fitting errors that are less than 0.3 gray levels.⁸

The algorithm requires labeling of points as purely diffuse or partially specular, which is done by using the degree of polarization ρ . Since ρ depends on I_{\min} , the noise level in ρ varies with I_{\min} . Rather than using a fixed threshold T_1 on ρ to identify partially specular points, our implementation uses a threshold that varies with I_{\min} . T_1 varies from around 5% for points with $I_{\min} > 200$ to around 10% for points when $I_{\min} \approx 20$.

The algorithm has three other thresholds that affect its performance; the threshold T_2 for the angle β , the threshold T_3 for the angle γ , and T_4 for the standard deviation σ_p . The algorithm is not too sensitive to T_2 and T_4 , which have been set at 0.08 and $\arccos(0.99)$, respectively. The angle threshold T_3 , which determines if points lie close to the plane \mathcal{P}_1 , has a strong affect on the quality of computed results as well as the computation time. The current implementation starts with a relatively small threshold value ($T_3=0.02$), and doubles it after every 10 iterations.

6.2 Experimental Results

In the each of the following examples, the images obtained after fitting the polarization parameters are used to compute the diffuse color image I_d . This image is then subtracted from the average color image I_{avg} to obtain the specular color image I_s . I_{avg} is the color image we would obtain if the polarization filter were not used. The first example is shown in Color Figure 1(a). It is the I_{avg} image of a mug with a flowered pattern on it. The petals of the flower are of different colors and within each petal there is a moderate amount of diffuse color variation. Along the middle of the mug is a large highlight. Color Figure 1(c) shows the I_{\min} image obtained after the polarization fitting process. This image represents the best image (i.e. minimum specular component) obtainable by simply rotating the

polarization filter. Color Figure 1(b) shows the diffuse image I_d computed using the proposed algorithm. Color Figure 1(d)–1(g) show close-up views of the I_{avg} diffuse image, specular image, and I_{\min} respectively, for a 40X40 image window on the flower cup. Clearly, the algorithm was successful in separating the two reflection components despite the texture underlying the highlight region.

The second example includes strong interreflection effects. Color Figure 2(a) shows the original image (I_{avg}) of a scene including a blue plastic plate and a part of the McBeth color chart. Color patches on the chart are reflected by the plate. There are pieces of plastic tape (some dark reddish and others black) stuck on the plate. Also visible is a film canister, which interreflects portions of the color chart as well as the surrounding environment. Color Figure 2(b) and Color Figure 2(c) show the diffuse and specular components computed by the algorithm. We see that, despite the strong interreflections, these images are quite accurate. We do, however, see that the primary highlight on the left side of plate has not been completely removed. This may have been caused by very high brightness values in the highlight region for which the sensor calibration is not reliable.

The final example is shown in Color Figure 3. Color Figure 3(a) shows the I_{avg} image of a complex scene for which the algorithm partially fails in some regions. Color Figure 3(b) and 3(c) show the diffuse and specular components computed by the algorithm. The primary highlights on the blue and red tori are accurately removed. Color Figure 3(d) – 3(f) show details of the results obtained for the blue torus. The diffuse and specular components are well separated in the highlight region. Errors in the separation are however seen on the occluding boundary of the torus. This results from the strong polarization of the diffuse component on the occluding boundary; the assumption that the diffuse component is unpolarized is violated.

Some of the interreflections on the left wall of the red box, such as the interreflection of the white cup in the upper left corner of the scene, are also removed. However, the results are not as good for the interreflections of the blue torus and

⁸ If the data is not consistent with the cosine model, one or more of the assumptions made by the algorithm are violated. Hence, pixels whose RMSE value is greater than 6 gray levels are marked as outliers and are not used for specular removal. For reasons mentioned above, fitting near scene edges is generally unreliable and hence errors in the specular removal occur mainly within 2-3 pixels around an edge.

the marker pen. This is because of the following reasons. First, the diffuse red color in these interreflection areas is very bright, resulting in a low (3-4 %) degree of polarization. Hence, many of the pixels are incorrectly labeled as "diffuse." Second, because of the geometry of the scene, the walls of the box interreflect between themselves, producing specular reflections that have the same color (red) as the diffuse component. As a result, the specular and diffuse colors could not be separated even when the algorithm thresholds were lowered to force a computation. Hence, it is not that the algorithm has removed too much red in the interreflections of the blue torus and the pen, but rather it was unable to remove the red specular reflection from the red wall. This is a manifestation of the dichromatic reflectance assumption made by the algorithm.

References

- [Bajcsy, et. al., 90] R. Bajcsy, S. W. Lee, and A. Leonardis "Color image segmentation with detection of highlights and local illumination induced by interreflections," *Proc. of International Conf. on Pattern Recognition*, Atlantic City, NJ, June 1990.
- [Born and Wolf, 65] M. Born and E. Wolf, *Principles of Optics*, London:Pergamon, 1965.
- [Boult and Wolff, 91] T. Boult and L.B. Wolff, "Physically based edge labeling," *Proc. CVPR*, pp. 656-663 1991.
- [Gershon, 87] R. Gershon, *The Use of Color in Computational Vision*, PhD thesis, University of Toronto, 1987.
- [Marvin, 88] *Handbook of Laser Science and Technology*, edited by M.J. Marvin, CRC Press Inc., Vol. 4, 1988.
- [Driscoll, 78] *Handbook of Optics*, edited by W.G. Driscoll, McGraw Hill Inc., 1978.
- [Klinker, 88] G. J. Klinker, *A Physical Approach to Color Image Understanding*, PhD thesis, Carnegie Mellon University, 1988.
- [Koshikawa, 79] K. Koshikawa, "A polarimetric approach to shape understanding," *Proc. IJCAI*, pp. 493-495, 1979.
- [Nayar, et. al., 91] S. K. Nayar, K. Ikeuchi, T. Kanade, "Surface Reflection: Physical and Geometrical Perspectives," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 7, pp. 611-634, July 1991.
- [Nayar et al., 1993] S.K. Nayar, X.S. Fang, and T.E. Boult. Separation of reflection components using color and polarization. Technical Report CUCS-58092, Columbia University Center for Research in Intelligent Systems.
- [Novak and Shafer, 92] C. Novak and S. Shafer, "Anatomy of a Color Histogram," *Proc. CVPR*, pp. 599-605, 1992.
- [Shafer, 85] S. Shafer, "Using color to separate reflection components," *Color Research and Applications*, Vol. 10, pp. 210-218, 1985.
- [Lee, 91] S.W. Lee, *Understanding of Surface Reflection in Computer Vision by Color and Multiple Views*, PhD thesis, University of Pennsylvania, 1991.
- [Torrance and Sparrow, 67] K. Torrance and E. Sparrow, "Theory for Off-Specular Reflection from Roughened Surfaces," *Journal of the Optical Society of America*, No. 57, pp. 1105-1114, 1967.
- [Wolff and Boult, 91] L.B. Wolff and T. Boult, "Constraining Object Features using a Polarization Reflectance Model," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 7, pp. 635-657, July 1991.
- [Wolff, 90] L.B. Wolff, *The Polaris System*. PhD thesis, Columbia University, 1990.

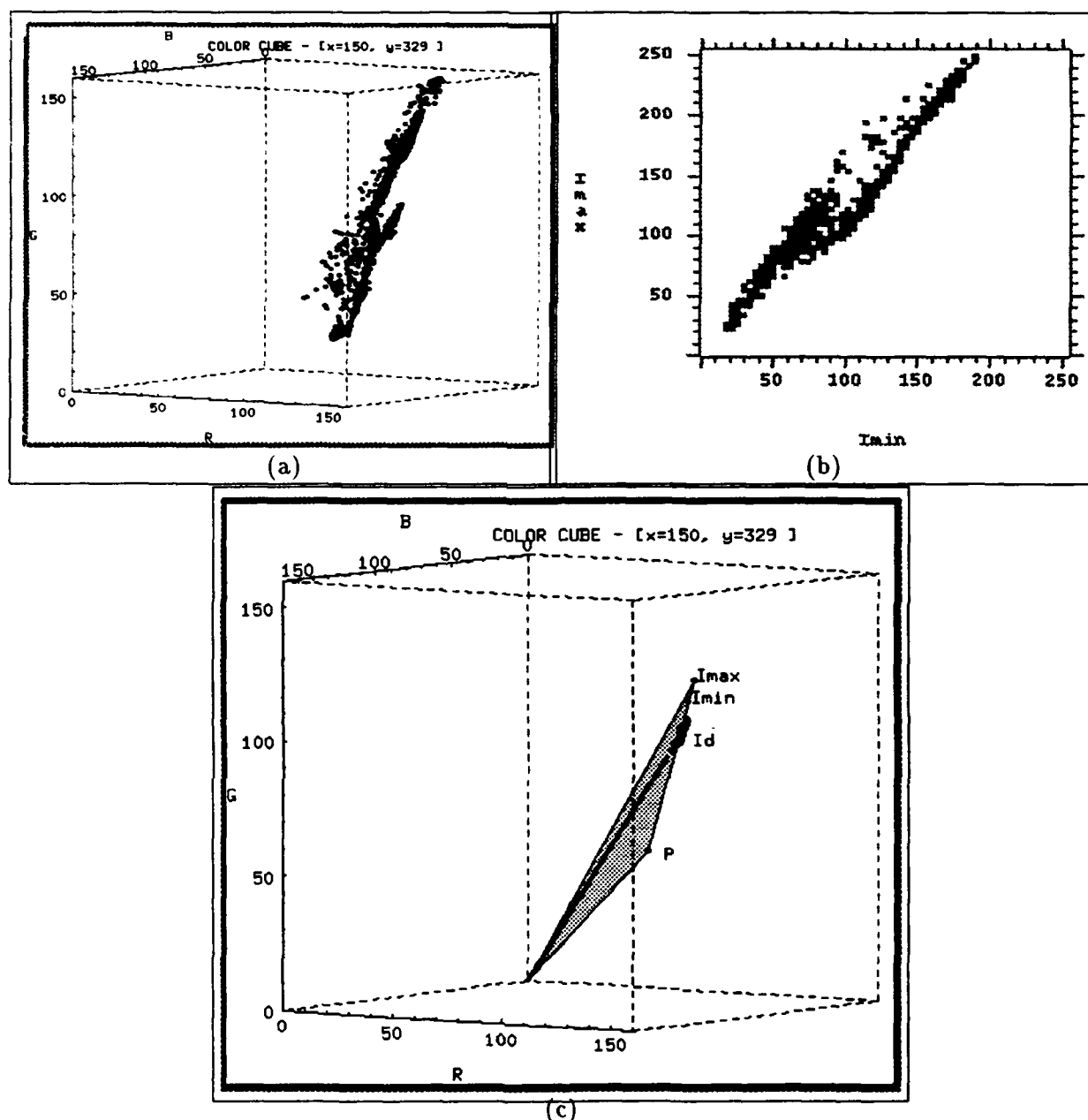


Figure 6: (a) The color histogram for a 40X40 window of the cup image shown in Color Figure 1(d). The anatomy of the histogram is too complex to identify the skewed T used in [Klinker, 88]. (b) Cluster in (I_{min}, I_{max}) space used by the polarization based proposed in [Wolff and Boulton, 91]. This cluster does not form a straight line, an assumption that the previous method is based on. (c) Separation of diffuse and specular components of the center pixel of the window using the proposed method based on color and polarization.

Section XVII

Low-Level Vision

Local Step Edge Estimation: a New Algorithm, Statistical Model and Performance Evaluation

Sheng-Jyh Wang, Thomas O. Binford *

Robotics Laboratory, Computer Science Department,
Stanford University, Stanford, CA 94305, U.S.A.

Abstract

This research is intended to enable recognition in real operational scenes with multi-sensor images by implementing generic image elements determined from the physics of image formation. Segmentation aims at a piecewise composite description of the image surface, i.e. smooth surfaces bounded by discontinuities.

A major failing is description of extended boundaries of the image surface. Estimation of extended edges is limited by accuracy of local edge estimation (edgel) in angle and transverse position. Angular accuracy of usual edgel estimation is so bad that it is usually ignored. A new operator with improved accuracy has led to preliminary extended edges that are greatly improved.

To incorporate image features in Bayesian inference, a statistical performance model is essential to provide conditional probabilities essential for evidential accrual. Theoretical models have been derived; they are verified by truth from synthesized imagery. For a previous operator, results were verified from truth in one image. Preliminary results have been obtained for other common edge estimation operators.

1. Introduction

This research is intended to enable recognition in real operational scenes with multi-sensor images by implementing generic image elements determined from the physics of image formation. Segmentation is common across many types of images in the sense that image formation involves only a few physical mechanisms from visible to IR: surface reflection, body reflection, emission. The role of segmentation is to describe the image surface compactly as a piecewise smooth composite surface, i.e. smooth surfaces bounded by discontinuities.

*This research was supported in part by a contract from the Air Force, F30602-92-C-0105 through RADC from DARPA SISTO, "Model-based Recognition of Objects in Complex Scenes: Spatial Organization and Hypothesis Generation".

With range images, a piecewise image surface description is directly a piecewise description of the visible object surface. With intensity images, a piecewise image description is a step toward inference of a piecewise object surface description.

Image surface segmentation generates piecewise smooth surfaces bounded by observable intensity discontinuities that result from discontinuities of surface geometry, of illumination, or reflectance. The intensity surface may have discontinuities at a point or along a curve. Discontinuities along curves are edges of the image intensity surface. Edges of the image surface are typically found by a local-to-global process of local estimation of discontinuities followed by hierarchical linking into extended boundaries. Complexity of linking is exponential in error of edgel orientation and transverse position. Errors of stereo depth estimates or dimensional measurements made from image extended curves are linear in errors of edgel orientation and position.

Little progress has been made on linking of extended boundaries. Although a display of edgels can be interpreted by a human observer, the set of edgels are unconnected and are not extended edges; only the human perceives extended edges that enable interpretation. In this paper, a new Wang-Binford edgel estimator is discussed which has led to effective edgel linking with a preliminary algorithm.

Almost all work on local segmentation of image discontinuities has been intended for step discontinuities between level, planar image surfaces. For typical images, only about a third or less of discontinuities are of this limited class. Typical edgel estimation algorithms are sensitive to shading, both in estimating spurious step edges where there are none, and in making inaccurate estimates at edges because of biases that result from an inaccurate model. Some workers ignore edgel angle information in using edge data. The algorithm described here makes more accurate estimates in the presence of shading.

[Herskovits and Binford, 1970] point out several types of discontinuities of the image surface are common in real images, corresponding to step, delta and crease ("roof") defined along curves. Also, discontinuities at points (spots) are important. This research has made an algorithm for edgel estimation for delta function discontinuities along curves, not described here.

The local-to-global structure is motivated by complexity, i.e. subdividing to describe small disks by simple functions that are computationally tractable. The edgel describes a step with parameters that are: a) orientation; b) transverse position; and c) contrast.

The Roberts cross and Canny operators are based on the gradient of intensity. The Canny operator is a one-dimensional operator with a heuristic extension to two dimensions. Both have bias in orientation and position in the presence of shading. The Binford-Horn operator was approximately a directional second-derivative.

For recognition in complex scenes, sensor and information fusion is based on Bayesian inference that depends on conditional probabilities of data given hypotheses [Binford 87a]. An extensive statistical model of performance has been developed, with theoretical analysis verified by truth from synthesized images. For a previous version of the operator, the model was verified by truth from a real image.

2. Wang-Binford Operator

2.1 Definition of a Step Edge

The profile of an ideal step edge can be expressed as

$$S(x) = A_1 \cdot U_{-1}(x)$$

where A_1 is a scalar and $U_{-1}(x)$ is the unit step function defined as

$$U_{-1}(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}$$

But the intensity image is blurred by the optical system and the impulse response of the sensor. The blurred edge can be approximated by Gaussian convolution expressed as:

$$S_b(x) = A_2 \cdot (U_{-1}(x) \otimes G_b(x))$$

where $G_b(x)$ is a Gaussian function with zero mean and standard deviation σ_b , and \otimes represents the 1-dimensional convolution. This is shown in figure 1.

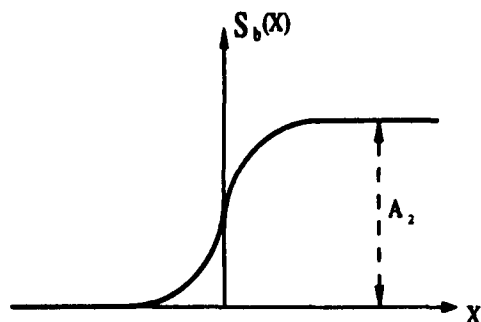


Figure 1. Profile of a Blurred Step Edge

Canny solved the one-dimensional variational problem of optimizing a combination of signal-to-noise and position location [Canny, 1983]. The result was approximately a gaussian derivative.

$$\begin{aligned} G_x(x, y) &= \frac{\partial}{\partial x} [I(x, y) * G_s(x, y)] \\ &= I(x, y) * \left[\frac{\partial}{\partial x} G_s(x, y) \right] \\ G_y(x, y) &= \frac{\partial}{\partial y} [I(x, y) * G_s(x, y)] \\ &= I(x, y) * \left[\frac{\partial}{\partial y} G_s(x, y) \right] \end{aligned}$$

where

$$G_s(x, y) = \frac{1}{2\pi\sigma_m^2} e^{-\frac{x^2+y^2}{2\sigma_m^2}}$$

is the 2-dimensional Gaussian smoothing function, and $*$ represents the 2-dimensional convolution.

Moreover, the 2-dimensional convolution can be decomposed into the product of two 1-dimensional Gaussian convolutions.

2.4 Detection and Localization

For a blurred step edge, the direction of its gradient is normal to the edge and the transverse profile of the gradient magnitude is a Gaussian function.

Without loss of generality, we can assume the step edge coincides with the x-axis. Its intensity function is expressed as

$$S(x, y) = A \cdot (U_{-1}(y) \otimes G_b(y))$$

After the gaussian derivative, the gradient is $(G_x(x, y), G_y(x, y))$, with

$$\begin{aligned} G_x(x, y) &= [S(x, y) \otimes \frac{\partial}{\partial x} G_m(x)] \otimes G_m(y) \\ &= [A(U_{-1}(y) \otimes G_b(y)) \otimes \frac{\partial}{\partial x} G_m(x)] \otimes G_m(y) \\ &= 0 \otimes G_m(y) \\ &= 0 \end{aligned}$$

and

$$\begin{aligned} G_y(x, y) &= [S(x, y) \otimes G_m(x)] \otimes \frac{\partial}{\partial y} G_m(y) \\ &= [A(U_{-1}(y) \otimes G_b(y)) \otimes G_m(x)] \otimes \frac{\partial}{\partial y} G_m(y) \\ &= [AU_{-1}(y) \otimes G_b(y)] \otimes \frac{\partial}{\partial y} G_m(y) \\ &= A \frac{\partial}{\partial y} [U_{-1}(y) \otimes G_b(y) \otimes G_m(y)] \\ &= A \frac{\partial}{\partial y} (U_{-1}(y) \otimes G_c(y)) \\ &= A \cdot G_c(y) \end{aligned}$$

where

$$G_c(t) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{t^2}{2\sigma_c^2}}$$

with $\sigma_c = \sqrt{\sigma_b^2 + \sigma_m^2}$.

However, if this edge is shaded, an extra term has to be added to the gradient. Since the shading effect is smooth, we can assume this extra term is constant within a local area, and the perturbed gradient function turns out to be

$$(G_x(x, y), G_y(x, y)) = (g_x, AG_e(y) + g_y)$$

As shown in Figure 2, the direction of the gradient vectors. If an edge detector depends on the gradient direction, like the Canny Operator does, one would expect it to have poor performance whenever shading is introduced. It is difficult to remove this extra component from the gradient if no information about the shading is known in advance.

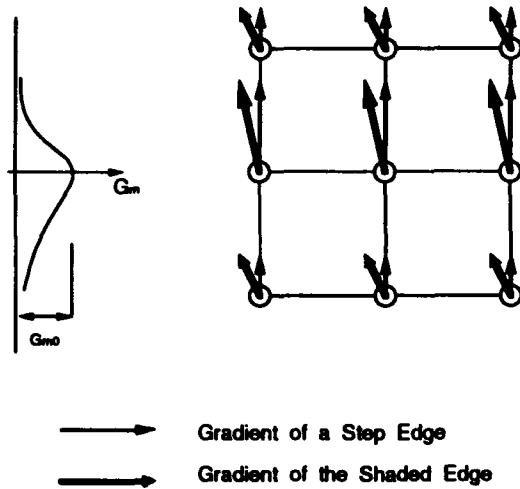


Figure 2. Gradients of a Shaded Step Edge

Even though the gradient vector is perturbed, the gradient magnitude function still keeps the information we need for unbiased localization. In the above case, its gradient magnitude function can be written as

$$G_m(x, y) = \sqrt{g_x^2 + (A \cdot G_e(y) + g_y)^2}$$

The peak of $G_m(x, y)$ coincides with the x-axis, that is, it coincides with the true edge. The gradient magnitude function is still constant along the x direction and is an even function in the y direction. The gradient magnitude extracts the orientation and position of the edge without bias. Moreover, even though the transverse profile of the gradient magnitude function is no longer a gaussian function, a gaussian function can still be used as a good approximation that will cause only a small bias in the estimation of the contrast, which is less important than bias in orientation and position.

To detect the edge, that is to detect the peak of the gradient magnitude function, the measured gradient magnitude values are fit with a parametric surface, which represents the gradient magnitude function of a step edge with parameters θ , u (transverse position), and G_{m0} (The gradient magnitude value at the edge). The fitting is done with a 3 by 3 support. Edge detection is solution of:

$$\min_{\theta, u, G_{m0}} \sum_{i,j=-1}^1 (G_m(x+i, y+j) - G_{m0} \cdot e^{-\frac{(-\sin \theta \cdot i + \cos \theta \cdot j - u)^2}{2\sigma_e^2}})^2$$

where θ is the orientation of the edgel, u is the transverse position shift from the center pixel, and G_{m0} is proportional to the contrast of the edgel. (as shown in figure 3)

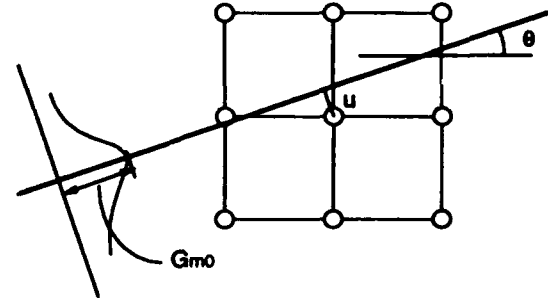


Figure 3. Parameters of a Blurred Step Edge

This is a nonlinear fitting problem. The σ_e , which depends on both σ_m and σ_b , is assumed to be known before the fitting. It can be estimated from the image easily. This nonlinear fitting problem can be even simplified to a linear one as follows.

The logarithm of the gradient magnitude of the blurred step edge is a quadratic:

$$\begin{aligned} \log(G_{m0} \cdot e^{-\frac{(-\sin \theta \cdot i + \cos \theta \cdot j - u)^2}{2\sigma_e^2}}) \\ = \log(G_{m0}) - \frac{(-\sin \theta \cdot i + \cos \theta \cdot j - u)^2}{2\sigma_e^2} \\ = ax^2 + bxy + cy^2 + dx + ey + f \end{aligned}$$

with

$$\begin{cases} a = -\frac{1}{2\sigma_e^2} \cdot \sin^2 \theta \\ b = \frac{1}{\sigma_e^2} \cdot \sin \theta \cdot \cos \theta \\ c = -\frac{1}{2\sigma_e^2} \cdot \cos^2 \theta \\ d = -\frac{1}{\sigma_e^2} \cdot \sin \theta \cdot u \\ e = \frac{1}{\sigma_e^2} \cdot \cos \theta \cdot u \\ f = \log(G_{m0}) - \frac{1}{2\sigma_e^2} \cdot u^2 \end{cases}$$

Therefore, a linear fit minimizes the objective function

$$\sum_{i,j=-1}^1 (\log G_m(x+i, y+j) - (ai^2 + bij + cj^2 + di + ej + f))^2$$

and the parameters of the edgel can be obtained from the coefficients of the parabolic function with

$$\begin{cases} \theta = \tan^{-1} \left(\frac{c-a+\sqrt{(a-c)^2+b^2}}{b} \right) \\ u = -\frac{-d \sin \theta + e \cos \theta}{a+c-\sqrt{(a-c)^2+b^2}} \\ \log(G_{m0}) = f - (a+c)u^2 \end{cases}$$

A linear solution gains efficiency and avoids convergence problems. It is not necessary to know σ , before the fitting. Logarithms are computed by table lookup. Thus, the Wang-Binford operator has computation cost comparable to the Canny operator.

2.5 Thresholding

Edges occur only at a small part of the image. Edge fitting is done only at local maxima of the gradient magnitude that are above threshold.

The threshold is determined by a constant false alarm rate (CFAR), that is, constant probability of detection of an edge where there is no edge actually. The variance of the gradient magnitude is determined from measurement of the variance of pixel intensity. The noise at pixel (i, j) , $n(i, j)$, is approximately an independent, identically distributed (i.i.d.) Gaussian random variable with zero mean, standard deviation σ_n . Its pdf can be expressed as

$$P_n(u) = \frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{u^2}{2\sigma_n^2}}$$

When there is no signal around (x, y) , the gradient becomes:

$$\begin{aligned} G_x(x, y) &= \sum_{i, j=-\infty}^{\infty} -\frac{i}{\sigma_m^2} \frac{1}{2\pi\sigma_m^2} e^{-\frac{i^2+j^2}{2\sigma_m^2}} n(x+i, y+j) \\ &\approx \sum_{i, j=-[4\sigma_m]}^{[4\sigma_m]} -\frac{i}{2\pi\sigma_m^4} e^{-\frac{i^2+j^2}{2\sigma_m^2}} n(x+i, y+j) \end{aligned}$$

and

$$\begin{aligned} G_y(x, y) &= \sum_{i, j=-\infty}^{\infty} -\frac{j}{\sigma_m^2} \frac{1}{2\pi\sigma_m^2} e^{-\frac{i^2+j^2}{2\sigma_m^2}} n(x+i, y+j) \\ &\approx \sum_{i, j=-[4\sigma_m]}^{[4\sigma_m]} -\frac{j}{2\pi\sigma_m^4} e^{-\frac{i^2+j^2}{2\sigma_m^2}} n(x+i, y+j) \end{aligned}$$

Since $G_x(x, y)$ and $G_y(x, y)$ are linear combinations of Gaussian random variables, they are again Gaussians. The pdf of the gradient $(G_x(x, y), G_y(x, y))$ is:

$$\begin{aligned} f_{grad_x, grad_y}(x, y) &= \frac{1}{2\pi \|A\|^{\frac{1}{2}}} e^{-\frac{\mathbf{r}^T A^{-1} \mathbf{r}}{2}} \\ &= \frac{1}{2\pi\sigma_g^2} e^{-\frac{x^2+y^2}{2\sigma_g^2}} \end{aligned}$$

where

$$\begin{cases} A = \begin{bmatrix} \sigma_g^2 & 0 \\ 0 & \sigma_g^2 \end{bmatrix} \\ \hat{V} = \begin{bmatrix} x \\ y \end{bmatrix} \\ \sigma_g^2 = \sum_{i, j} \frac{i^2}{4\pi^2\sigma_m^4} e^{-\frac{i^2+j^2}{2\sigma_m^2}} \sigma_n^2 \approx \frac{\sigma_n^2}{8\pi\sigma_m^4} \end{cases}$$

The pdf of the gradient magnitude is:

$$\begin{aligned} f_{gm}(x) &= \int_0^{2\pi} \frac{x}{2\pi\sigma_g^2} e^{-\frac{1}{2}\frac{x^2}{\sigma_g^2}} d\theta \\ &= \frac{x}{\sigma_g^2} e^{-\frac{1}{2}\frac{x^2}{\sigma_g^2}} \quad x \geq 0 \end{aligned}$$

If the false positive rate is set to be α , only $(\alpha \cdot 100)\%$ of gradient magnitude value at the background can be greater than threshold g_T . That is

$$\begin{aligned} 1 - \alpha &= \int_0^{g_T} \frac{x}{\sigma_g^2} e^{-\frac{1}{2}\frac{x^2}{\sigma_g^2}} dx \\ &= \int_0^{g_T} \frac{1}{2\sigma_g^2} e^{-\frac{1}{2}\frac{x^2}{\sigma_g^2}} (2x dx) \\ &= \int_0^{g_T^2} \frac{1}{2\sigma_g^2} e^{-\frac{1}{2}\frac{y}{\sigma_g^2}} dy \\ &= 1 - e^{-\frac{g_T^2}{2\sigma_g^2}} \end{aligned}$$

From the above equation, the threshold g_T has value:

$$g_T = [2\sigma_g^2 \ln(\frac{1}{1-\alpha})]^{\frac{1}{2}}$$

2.6 Details of Implementation

To compute the threshold for edge detection, the value of σ_n , the standard deviation of pixel intensity, is needed. It varies from image to image. It can be measured from the sensor with controlled scenes. It can be estimated from an existing image to a good approximation.

At any pixel of the image, the intensity value consists of signal with noise. To separate the noise from the signal, consider the approximation to the laplacian:

$$\begin{aligned} K(x, y) &= I(x, y) - \frac{1}{4}[I(x-1, y) + I(x+1, y) \\ &\quad + I(x, y-1) + I(x, y+1)] \end{aligned}$$

If the intensity surface of the signal approximates a plane locally, then $K(x, y)$ consists only of a noise term. With the assumption of i.i.d. noise, the variance of K becomes:

$$\begin{aligned} \text{Variance of } K(x, y) &= (1 + 4 \cdot (0.25)^2) \sigma_n^2 \\ &= 1.25 \cdot \sigma_n^2 \end{aligned}$$

That is,

$$\sigma_n = 0.8944 \sqrt{\text{Variance of } K(x, y)}$$

A digital image is sampled at discrete pixels. Both the impulse response function of camera pixels and of the Gaussian derivative contribute to discretization. The effect of discretization is large for small values of the width of the Gaussian derivative. This effect has been examined theoretically and experimentally.

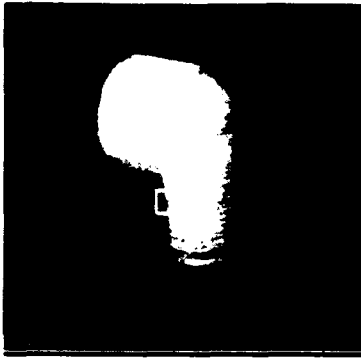


Figure 4.1.a Intensity Image

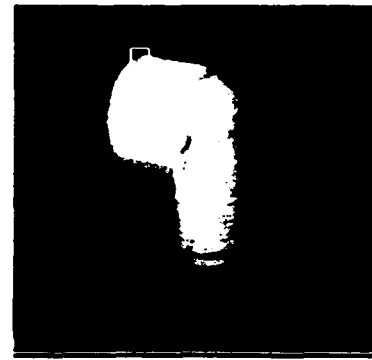


Figure 4.2.a Intensity Image

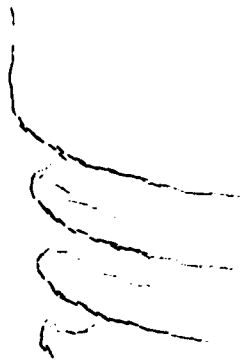


Figure 4.1.b Roberts Operator



Figure 4.2.b Roberts Operator



Figure 4.1.c Canny Operator



Figure 4.2.c Canny Operator

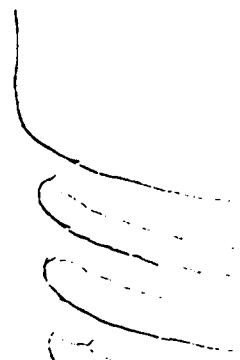


Figure 4.1.d Wang-Binford Operator



Figure 4.2.d Wang-Binford Operator

The σ_m of the smoothing filter has to be carefully chosen to avoid this effect. It has been claimed in [Canny, 1983] that the minimum value of the σ of the Gaussian smoothing function, which is applied before sampling, is approximately equal to the sampling interval τ . With a similar analysis, the value of σ_c , which depends on both σ_m and σ_b , has to be bigger than 1 to avoid the discretization effect. This conclusion is supported by empirical data.

3. Statistical Model of the Detected Edgel

The first order statistical model of this operator is its bias vector. The second order statistical model is its variance. Together they specify a Gaussian probability model for the operator to use in performance analysis and in Bayesian inference.

Noise in the intensities of image pixels causes the position, orientation, and contrast of the estimated edgel to fluctuate. This statistical model can be estimated by either experiment or theory. We did it in both ways. It is found that the pdf of the fluctuation can be approximated by a 3-dimensional Gaussian distribution, with mean vector $\mathbf{m} = \mathbf{0}$ and diagonal covariance matrix Λ .

3.1 Experimental Method

A program was written to generate intensity images with known gaussian impulse response function of a step edge with given position, orientation, and contrast. Random noise with given energy σ_n^2 , was added to image values and the fluctuation of θ , u , and $\log(G_{m0})$ was measured. This process is repeated 5000 times to estimate the covariance for orientation, position and gradient as a function of edge contrast.

Orientations were 0, 11.25, 22.5, 33.75, and 45 degrees, and contrast was 4, 8, 16, 32, 64, 128, and 256. The result of this experimental approach shows that the pdf of the fluctuation is approximated by a 3-dimensional Gaussian distribution with mean $\mathbf{m} = \mathbf{0}$ and diagonal covariance Λ . The model is insensitive to orientation, while the standard deviation of the fluctuation is proportional to $\frac{\sigma_n}{\text{contrast}}$ as shown in Figure 5.

A similar simulation was done for circular edges for radius 5, 10, 20, 40, 80, 160, and 1600 pixels. The result is about the same as the straight edge case when the radius is bigger than 10 pixels. When the radius is smaller than 10 pixels, one extra constant term has to be added to the standard deviation of $\log(G_{m0})$ as shown in Figure 6.

3.2 Theoretical Analysis

We use a perturbation method to get the analytic model as follows:

In detection, parameters of the edgel minimize the objective function:

$$\Phi(P, W) = \sum_{i,j=-1}^1 [\log(G_m(x+i, y+j)) - (ai^2 + bj + cj^2 + di + ej + f)]^2$$

where P stands for the coefficients a, b, c, d, e, f of the parabolic function, and W stands for the set of measured data $J(x, y) = \log(G_m(x, y))$.

That is, the parameters are solutions for P such that:

$$\Psi(P, W) = \frac{\partial \Phi(P, W)}{\partial P} = 0$$

Denote P_0 as the solution of $\Psi(P, W_0) = 0$, where W_0 stands for the noise-free case. When noise is added, it causes the measured data to perturb a small amount δW , which consequently causes a small fluctuation δP in the coefficient space. The relationship between δW and δP is given by:

$$\Psi(P_0 + \delta P, W_0 + \delta W) = 0$$

By expanding the above equation, we have

$$\frac{\partial \Psi}{\partial P} \cdot \delta P + \frac{\partial \Psi}{\partial W} \cdot \delta W \approx 0$$

and thus

$$\delta P \approx -\left(\frac{\partial \Psi}{\partial P}\right)^{-1} \cdot \frac{\partial \Psi}{\partial W} \cdot \delta W$$

Moreover, the parameter of the edgel, Θ , is a function of P with $\Theta = M(P)$. Thus:

$$\begin{aligned} \delta \Theta &= \frac{\partial M(P)}{\partial P} \delta P \\ &= -\left[\frac{\partial M(P)}{\partial P} \cdot \left(\frac{\partial \Psi}{\partial P}\right)^{-1} \cdot \frac{\partial \Psi}{\partial W}\right] \cdot \delta W \\ &= \Omega \cdot \delta W \end{aligned}$$

The 1st and 2nd moments of $\delta \Theta$ can be computed as:

$$\begin{aligned} E(\delta \Theta) &= \Omega \cdot E(\delta W) \\ E(\delta \Theta \delta \Theta^T) &= \Omega \cdot E(\delta W \delta W^T) \cdot \Omega^T \end{aligned}$$

Therefore, the statistical model of $\delta \Theta$ follows from the statistical model of δW .

3.2.1 Statistical Model for the Measured Data

With the linearity of convolution and the noise model above, the fluctuation of gradient is caused by noise only and is independent of signal. That is:

$$\delta G_x(x, y) = \iint -\frac{u}{2\pi\sigma_m^4} e^{-\frac{u^2+v^2}{2\sigma_m^2}} N(x-u, y-v) du dv$$

$$\delta G_y(x, y) = \iint -\frac{v}{2\pi\sigma_m^4} e^{-\frac{u^2+v^2}{2\sigma_m^2}} N(x-u, y-v) du dv$$

where $N(i, j)$ is the noise component at pixel (i, j) . It has been assumed to be an i.i.d. Gaussian random variable with zero mean, and constant standard deviation σ_n .

Since integration is a linear operation, $\delta G_x(x, y)$ and $\delta G_y(x, y)$ are also Gaussians. With some straightforward computations, we have:

$$E(\delta G_x(x, y)) = E(\delta G_y(x, y)) = 0$$

and

$$\begin{aligned}
E(\delta G_x(x, y) \delta G_x(\hat{x}, \hat{y})) \\
&= \sigma_n^2 \left[\frac{1}{8\pi\sigma_m^4} - \left(\frac{x - \hat{x}}{2} \right)^2 \frac{1}{4\pi\sigma_m^6} \right] e^{-\frac{1}{4\sigma_m^2}[(x - \hat{x})^2 + (y - \hat{y})^2]} \\
&= \sigma_n^2 \cdot N_{xx}(x - \hat{x}, y - \hat{y})
\end{aligned}$$

$$\begin{aligned}
E(\delta G_x(x, y) \delta G_y(\hat{x}, \hat{y})) \\
&= -\sigma_n^2 \frac{1}{16\pi\sigma_m^6} (x - \hat{x})(y - \hat{y}) e^{-\frac{1}{4\sigma_m^2}[(x - \hat{x})^2 + (y - \hat{y})^2]} \\
&= \sigma_n^2 \cdot N_{xy}(x - \hat{x}, y - \hat{y})
\end{aligned}$$

$$\begin{aligned}
E(\delta G_y(x, y) \delta G_y(\hat{x}, \hat{y})) \\
&= \sigma_n^2 \left[\frac{1}{8\pi\sigma_m^4} - \left(\frac{y - \hat{y}}{2} \right)^2 \frac{1}{4\pi\sigma_m^6} \right] e^{-\frac{1}{4\sigma_m^2}[(x - \hat{x})^2 + (y - \hat{y})^2]} \\
&= \sigma_n^2 \cdot N_{yy}(x - \hat{x}, y - \hat{y})
\end{aligned}$$

Now, investigate the dependence of $\log(G_m(x, y))$ with noise. By definition:

$$J(x, y) = \log(G_m(x, y)) = \log(\sqrt{G_x(x, y)^2 + G_y(x, y)^2})$$

Therefore,

$$\delta J(x, y) = \frac{G_x(x, y) \delta G_x(x, y) + G_y(x, y) \delta G_y(x, y)}{G_x(x, y)^2 + G_y(x, y)^2}$$

Again, since $\delta J(x, y)$ is the linear combination of two Gaussian random variables, $\delta G_x(x, y)$ and $\delta G_y(x, y)$, it is also a Gaussian random variable with:

$$\begin{aligned}
E(\delta J(x, y)) &= 0 \\
E(\delta J(x, y) \delta J(\hat{x}, \hat{y})) \\
&= \frac{\sigma_n^2}{G_{m0}^2} Q(x - \hat{x}, y - \hat{y}) \\
&\quad \cdot \int_{-0.5}^{0.5} e^{\frac{(-\sin \theta x + \cos \theta y - u)^2 + (-\sin \theta \hat{x} + \cos \theta \hat{y} - u)^2}{2\sigma_n^2}} du \\
&= \frac{\sigma_n^2}{G_{m0}^2} H(x, y, \hat{x}, \hat{y})
\end{aligned}$$

where:

$$\begin{aligned}
Q(u, v) &= \sin^2 \theta N_{xx}(u, v) - 2 \sin \theta \cos \theta N_{xy}(u, v) \\
&\quad + \cos^2 \theta N_{yy}(u, v)
\end{aligned}$$

3.2.2 Statistical Model for θ , u , and $\log(G_{m0})$

The statistical model of δW combined with the knowledge of $\frac{\partial M(P)}{\partial P}$, $(\frac{\partial \Psi}{\partial P})^{-1}$, and $\frac{\partial \Psi}{\partial W}$ gives the statistical model of $\delta \Theta$. The result shows that the pdf of $[\delta \theta, \delta u, \delta(\log(G_{m0}))]$ can be approximated by a 3-dimensional Gaussian distribution with zero mean vector and diagonal covariance. Moreover, the standard deviations of $\delta \theta$, δu , and $\delta(\log(G_{m0}))$ are found to be proportional to σ_n/G_{m0} .

These two models, from synthesized data and from the theoretical analysis agree for a straight step edge and for circular arcs with radius of curvature greater than 10 pixels. (see Figure 5.) When the radius is smaller than 10 pixels, a constant term, $\delta(\log(G_{m0}))$, has to be added to the standard deviation, as shown in Figure 6 and estimates of u and $\log(G_{m0})$ are slightly biased.

A more complicated statistical model for δW has to be made for edges with large curvature. This part is not finished yet. However, the model for the straight edge is still good enough for most of the image.

4. Conclusion

In this paper, we demonstrated an algorithm for detection and estimation of step edges which is insensitive to shading. Even though this operator is designed especially for straight step edges, it can also be applied to most curved edges. The detected edgels are unbiased in orientation and position. The resulting improved accuracy enables linking into extended edges and improved accuracy of estimation of image dimensions.

The statistical edgel model was constructed and verified. This model is valuable in the Bayesian network for combining evidence. Even though this model is good enough for most images, we are planning to build a more complete model for highly curved step edges.

This operator is still not effective for many image features, e.g. lines or spots. A complete set of operators has to be developed to deal with other basic image features.

References

- [Canny, 1983] Canny, J.F., "Finding Edges and Lines in Images", M.I.T., Cambridge, Tech. Rep. 720, 1983.
- [Herskovits and Binford, 1970] Herskovits, A. and Binford, T.O. "On Boundary Detection", M.I.T. Artificial Intelligence Lab., Cambridge Mass, AI Memo 183, 1970.
- [Leipnik, 1960] Leipnik, R. "The extended entropy uncertainty principle", Information and Control 3, 18-25, 1960.
- [Marr and Hildreth, 1980] Marr D.C. and Hildreth E.C., "Theory of edge detection", Proc. Roy. Soc. London B, vol 204, 1980.
- [Roberts, 1968] Roberts L.G., "Machine Perception of Three-Dimensional Solids", Optical and Electro-Optical Information Processing, 159-197, 1968.
- [Torre and Poggio, 1986] Torre, V. and Poggio, T.A., "On Edge Detection", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. PAMI-8, 147-163, 1986.

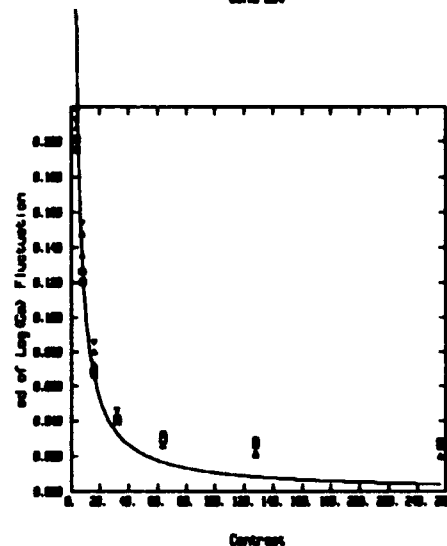
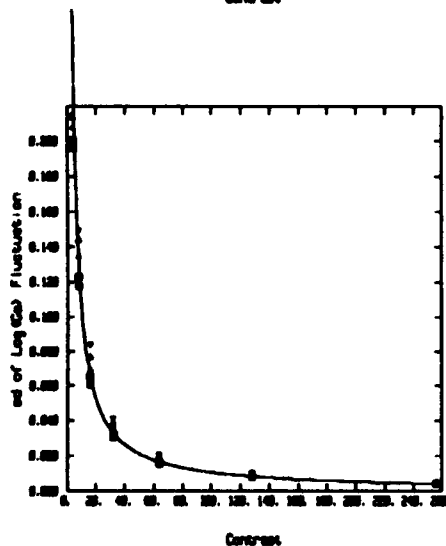
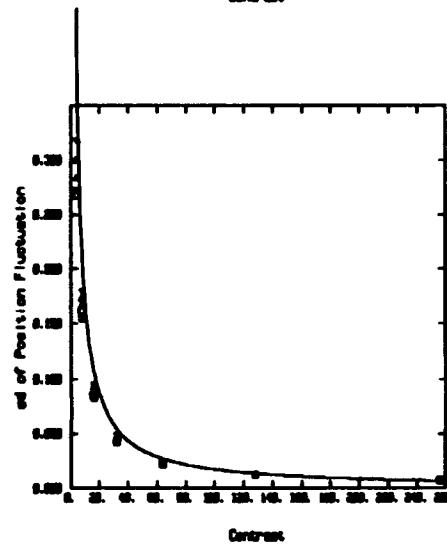
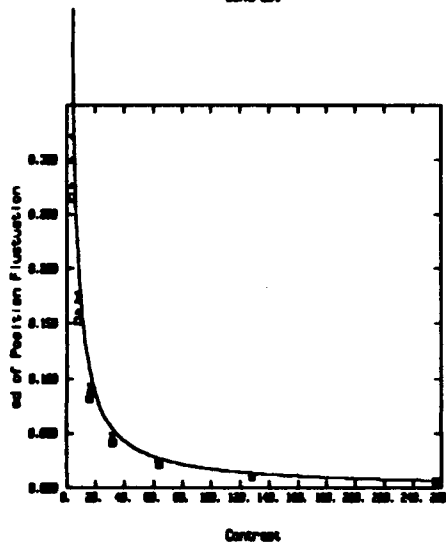
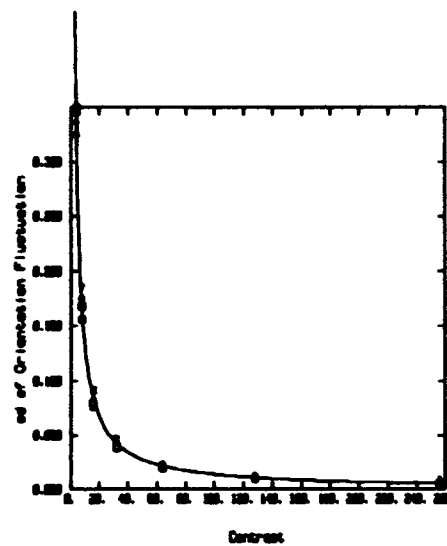
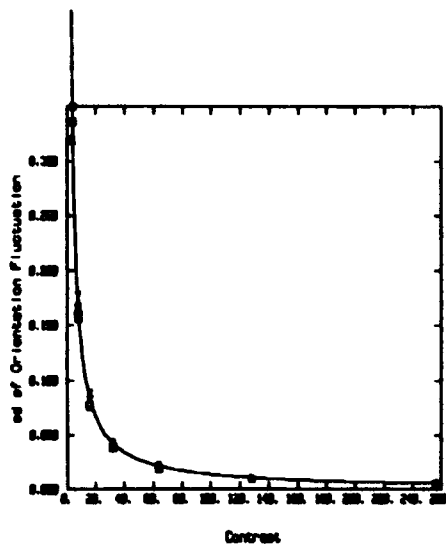


Figure 5. Statistical Model (Radius = 1600 pixels)
 (symbols : empirical data
 curve : theoretical model)

Figure 6. Statistical Model (Radius = 5 pixels)
 (symbols : empirical data
 curve : theoretical model)

Performance Characterization of Edge Operators

Visvanathan Ramesh *

Robert M. Haralick

Department of EE, FT-10

University of Washington

Seattle WA 98195

Abstract

Computer vision algorithms are composed of different sub-algorithms often applied in sequence. Determination of the performance of a total computer vision algorithm is possible if the performance of each of the sub-algorithm constituents is given. The performance characterization of an algorithm has to do with establishing the correspondence between the random variations and imperfections in the output data and the random variations and imperfections in the input data. In the paper by Ramesh and Haralick, [1], theoretical models for the random perturbations in the output of a vision sequence, involving edge finding, edge linking and line fitting were presented. They modelled the process that describes the breakage of a true model line segment by a renewal process with alternating line and gap intervals. However, their paper assumed independence of gradient estimates obtained from neighboring pixel locations.

In this paper we show how one can relax the independence assumptions and derive perturbation models that include the effects of correlation between neighboring gradient estimates. Under the assumption that the ideal data is corrupted with additive, independent additive gaussian noise, we derive expressions that describe the relationship between an edge gradient estimate at a given location and an edge gradient estimate for a neighboring pixel. We illustrate how the model line breakage process can be modeled as a Markov process whose parameters are functions

of the true edge gradient, edge operator's neighborhood size, and the noise variance. Furthermore, we derive theoretical expressions for the mean positional error as a function of the neighborhood operator window size, noise variance, the width of the true ramp edge, and the true edge gradient. We also outline an experimental protocol used for evaluating edge pixel positioning errors and discuss the results obtained from the experiments.

1 Introduction

Computer vision algorithms are composed of different sub-algorithms often applied in sequence. Determination of the performance of a total computer vision algorithm is possible if the performance of each of the sub-algorithm constituents is given. The performance characterization of an algorithm has to do with establishing the correspondence between the random variations and imperfections in the output data and the random variations and imperfections in the input data. In the paper by Ramesh and Haralick, [1], theoretical models for the random perturbations in the output of a vision sequence, involving edge finding, edge linking and line fitting were presented. They modelled the process that describes the breakage of a true model line segment by a renewal process with alternating line and gap intervals. However, their paper assumed independence of gradient estimates obtained from neighboring pixel locations.

In this paper we show how one can relax the independence assumptions and derive perturbation models that include the effects of correlation between neighboring gradient estimates. Under

*This project has been funded in part through a contract from DARPA. Funding for V.Ramesh from IBM in the form of a IBM Manufacturing Research Fellowship is also gratefully acknowledged

the assumption that the ideal data is corrupted with additive, independent additive gaussian noise, we derive expressions that describe the relationship between an edge gradient estimate at a given location and an edge gradient estimate for a neighboring pixel. We illustrate how the model line breakage process can be modeled as a Markov process whose parameters are functions of the true edge gradient, edge operator's neighborhood size, and the noise variance. Furthermore, we derive theoretical expressions for the mean positional error as a function of the neighborhood operator window size, noise variance, the width of the true ramp edge, and the true edge gradient. This paper is organized as follows. The first section provides a brief review of results discussed in [1]. The second section provides an analysis of the positional error introduced by gradient based edge operators. The third section provides a discussion of a perturbation model for the edge output that takes into account the dependence of estimates at neighboring pixels. A subsequent section provides a discussion of theoretical performance measure plots.

2 Review of previous results

In this section we review some of the results outlined in the paper by Ramesh and Haralick [1]. Our work extends the results given in [1]. Ramesh and Haralick [1], describe a theoretical model by which pixel noise can be successively propagated through an edge labelling algorithm, an edge linking algorithm and a boundary gap filling algorithm. Assuming an edge idealization of a linear ramp edge and i.i.d Gaussian random perturbations on pixel grayvalues they show how one could model the breakage of a true line segment as a renewal process with alternating segment and gap intervals. They show that if one ignores the dependencies between adjacent gradient estimates then the segment and gap interval lengths are exponentially distributed with parameters λ_1 and λ_2 that are related to the true edge gradient, the neighborhood operator size and the gradient threshold employed. They also show how the output after a gap filling operation could still be modeled as an alternating renewal process and derive the length distributions for the segment and gap intervals after

the operation.

In reality, there is an overlap between the edge detector neighborhoods centered around pixels and hence there is some dependence between gradient estimates obtained for neighboring windows. In addition, if one assumes that the noise at each pixel is locally dependent then the correlation in the noise would introduce correlation in the gradient estimates. In addition, the analysis in [1] did not include positional errors. These positional errors are of significance if one wishes to analyze higher-level matching algorithms.

In other work, [2], we focussed on performing theoretical model-based comparison of gradient based edge finding schemes and mathematical morphology based edge finding schemes. The performance analysis was done by assuming an ideal edge model and a noise model and by deriving expressions for probability of false alarm and probability of misdetection of edge pixels. Under the Gaussian noise model assumption, the theory indicated that the morphological edge detector is superior to conventional gradient based edge detectors, that label edges based on gradient magnitude, when a size 3 by 3 window was used. We performed experiments to validate our theoretical results and the empirical plots indicated that the morphological edge operator was also superior when a 5 by 5 window is used. However, the theoretical plots did not confirm this because the theory provided only an upperbound. In [2] we also included comparisons of results obtained for real images. A simple analysis of hysteresis linking was also done in this paper and it was shown that hysteresis linking improves the performance of the edge operators.

3 Positional Error Analysis of Gradient Based Edge Detectors

In this section we derive the expression for the mean error in the edge pixel location. We consider an ideal edge model that has an one-dimensional intensity profile of a ramp. Specifically, the intensity profile is defined by:

$$I(x) = a + Gx \quad (1)$$

for $x = -K - 1/2, \dots, K - 1/2$

$$\begin{aligned}
&= a - G(K-1)/2 \text{ for } x < -K-1/2 \\
&= a + G(K-1)/2 \text{ for } x > K-1/2
\end{aligned}$$

We assume that the edge detection is performed by computing the gradient by fitting a planar surface to the grayscale values as in [1]. In 1-dimension this problem is equivalent to fitting a line to the data for each 1 by K neighborhood. There are two kinds of errors that are introduced in the fit, one error is the systematic bias that is introduced in the fit due to the approximation of the function $I(x)$ by a linear fit in the 1 by K neighborhood and the other error is the error introduced due to the additive noise in the input. Let $G(x)$ be the gradient estimate obtained when the least squares fit is performed for the window of ideal data $I(i), i = x - (K-1)/2, \dots, x + (K-1)/2$. Clearly, $G(0)$, the gradient estimate when the neighborhood overlaps the entire ramp, is equal to the true slope G . Also, $G(x), |x| > K$ is equal to zero. In addition, one can note that $G(x)$ is a symmetric function since $I(x)$ is symmetric. When the discrete samples are corrupted with additive i.i.d Gaussian noise with zero mean and variance σ^2 , then the estimates for the gradient values, $\hat{G}(x)$, are normal random variables with true mean $G(x)$ and variance $\sigma^2 / \sum i^2$ where the sum is taken over values of $i = -(K-1)/2, \dots, (K-1)/2$. Neighboring gradient estimates, $\hat{G}(x)$ and $\hat{G}(x+j)$, are dependent random variables because of the overlap in the neighborhoods used during the estimation procedure.

We show in the appendix that if we viewed the sequence of $2K-1$ random variables $\hat{G}(x), x = -(K-1), \dots, K-1$ as a random vector \hat{G} then \hat{G} is distributed as a multivariate Gaussian random variable with mean vector $G(x)$ and covariance matrix $A \Sigma A'$, where the matrix A is obtained from fitting kernel coefficients as described in the appendix and Σ is the covariance matrix of the additive noise vector which is assumed to be $\sigma^2 I$. The matrix A captures the dependence between the adjacent gradient estimates.

In order to compute the error in the edge pixel position, we assume that the pixel with the maximum gradient magnitude along the gradient direction is labelled as an edge, while all the other pixels are labelled as non-edge pixels. That

is, the edge pixel's index is ep when:

$$\begin{aligned}
&\hat{G}(ep) > \hat{G}(x) \\
&\forall x > -(K-1), x < (K-1), x \neq ep
\end{aligned} \tag{2}$$

Hence the probability that the location i is labelled as edge is given by a multivariate integral with appropriate limits specified by the gradient threshold used. That is, the probability is given by the expression:

$$\begin{aligned}
Prob(ep = i) = & \int_{x_i=T}^{\infty} \int_{x_j=0}^{x_i} \dots \int_{j \neq i} \Phi(G(x), A \Sigma A') dx_i dx_j \\
& + \int_{x_i=-\infty}^{-T} \int_{x_j=x_i}^0 \dots \int_{j \neq i} \Phi(G(x), A \Sigma A') dx_i dx_j
\end{aligned} \tag{3}$$

where Φ is the multivariate normal distribution function. Φ has two sums in the integral because the threshold T is actually on the absolute value of the gradient. The mean error in the edge pixel location is then given by:

$$\mu = \sum_{i=0}^{K-1} i Prob(ep = i) \tag{4}$$

4 Boundary Model incorporating Dependencies in Estimates

In the previous section we addressed how errors in grayvalues propagate to errors in pixel locations at the output of the edge operator. An alternating renewal process with gaps and edge segments ([1]) is used to describe the breakage of a true model segment into short edge segments and gaps. Under the assumption that the gradient across the edge is constant along the true model boundary and ignoring the dependence between gradient estimates from local neighbours, it was seen in [1] that the edge segment lengths and the gap lengths can be approximated as exponential distributions. In this section we illustrate how the boundary model given in [1] can be extended to include the dependencies due to correlation of gradient estimates.

We assume the ideal model for the intensity profile across the boundary to be a ramp edge

with constant gradient as one walks along the model line. In addition we assume that the samples are corrupted with i.i.d. additive Gaussian noise. It is shown in the appendix how one can relate the gradient estimate obtained at a particular location, (r, c) , to the gradient estimate at a nearby location, $(r+k, c+j)$. Using these relationships one can derive the expression for the probability that the gradient estimate $\hat{G}(r+k, c+j)$ is greater than the threshold T given that the gradient estimate $\hat{G}(r, c) > T$. For simplicity, we can assume that the ramp edge is oriented across the column direction. In this situation we are interested in modelling the relationship between the gradient estimates at successive rows. This scenario is equivalent to the examination of the gradient estimates in neighboring pixels, as one walks along the true model line. We visualize the sequence of edge labels (1's and 0's) as we walk along the model line as a series of binary random variables. It is easily seen that if we are dealing with independent Gaussian noise at each pixel, the gradient estimate $\hat{G}(r, c)$ is dependent on the previous $(\hat{G}(r-k, c), k = 1, \dots, K)$ estimates. In this sense, the binary edge sequence forms a binary K th order Markov chain. The Markov chain can be specified by the conditional probabilities: $Prob(X_r = 1 | X_{r-k} = 1), k = 1, \dots, K$ and $Prob(X_r = 1 | X_{r-1} = 1, \dots, X_{r-j} = 1), j = 2, \dots, K$. These probabilities can be easily derived from the joint distribution for the $\hat{G}()$'s by computing the appropriate multivariate integral. For example: $Prob(X_r = 1 | X_{r-k} = 1)$ is equal to $Prob(\hat{G}(r, c) \geq T | \hat{G}(r-k, c) \geq T)$ and is given by: $Prob(\hat{G}(r, c) \geq T, \hat{G}(r-k, c) \geq T) / Prob(\hat{G}(r-k, c) \geq T)$. The numerator in the above expression is obtained by integrating the joint distribution of $\hat{G}(r, c)$ and $\hat{G}(r-k, c)$ with limits of integration from T to ∞ . The denominator is the integral of the distribution function for $\hat{G}(r-k, c)$.

5 Protocol for image generation (for edge pixel accuracy) and evaluation

Synthetic images of size 51 rows by 51 columns were generated with step edges at various orientations passing through the center pixel $(R, C) =$

$(26, 26)$ in the image. The gray value, $I(r, c)$, at a particular pixel, (r, c) , in the synthetic image was obtained by using the function where $\rho = (r - R)\cos(\theta) + (c - C)\sin(\theta)$.

$$\begin{aligned} I(r, c) &= I_{min}, \quad \rho < 0 \\ &= I_{max}, \quad \text{otherwise.} \end{aligned} \quad (5)$$

I_{min} and I_{max} are the gray values in the left and right of the step edge. The variables R and C designate a point in the image on which the step edge boundary lies. In our experiments we set I_{min} to be 100 and I_{max} to be 200. We used orientation (θ) values of 0, 15, ..., 175 degrees. To generate ramp edges, we averaged images containing the step edges with a kernel of size 4by4 so that the resulting ramps have 5 pixels width. To these ramp edge images we added additive Gaussian noise to obtain images with various signal to noise ratios. We define signal to noise ratio as:

$$SNR = 20 \log \left(\frac{\sigma_s}{\sigma_n} \right) \quad (6)$$

where σ_s is the standard deviation of the gray values in the input image and σ_n is the noise standard deviation. We used SNR values of 0, 5, 10, 20 dB. They correspond to σ_s/σ_n values of 1, 1.78, 3.162, and 10 respectively. Groundtruth edge images were generated by using the following function where $\rho = (r - R)\cos(\theta) + (c - C)\sin(\theta)$.

$$\begin{aligned} I_1(r, c) &= 0 \quad \rho < -0.5 \\ &= 1 \quad \text{otherwise.} \\ I_2(r, c) &= 0 \quad \rho < 0.5 \\ &= 1 \quad \text{otherwise.} \\ I(r, c) &= I_1(r, c) \text{ exor } I_2(r, c) \end{aligned} \quad (7)$$

The operators employed included the gradient based (Gradient computed using the slope facet model) operator and the morphological blur-minimum operator discussed in [3]. In the Blur-minimum morphological edge detector a pixel is assigned an edge label if the edge strength computed is above a given threshold T . The edge strength I_e is given by the equation:

$$I_e = \min\{I_1 - \text{erosion}(I_1, \text{disk}(r)), \text{dilation}(I_1, \text{disk}(r)) - I_1\} \quad (8)$$

where I_1 is the input image and r is the radius of the disk that is used as the structuring element

in the morphological erosion/dilation operations. We used 5 by 5 neighborhoods for the edge operator and the blur-minimum operator. The edge accuracy evaluation proceeded as follows. The edge pixel location error E is defined as the distance along the gradient direction from the true edge pixel to the nearest labelled edge pixel (if one exists, in the edge detector output). A given ground truth edge pixel is assumed to be missing in the detector output if there are no edge pixels in the detector output within an interval centered on the ground truth edge pixel. The interval is oriented along the gradient direction and the number of pixels in the interval is equal to the edge operator width. We will refer to this interval, as the "valid zone" for each pixel.

In addition to the computation of edge pixel location error as given above, we also compute the following statistics from the output image. We visualize the edge and non-edge labellings encountered as one walks along the valid zone as a sequence of alternating 0 and 1 runs. We compute the mean and variances for the lengths of the gaps and the edge segments. In the ideal case when there is no error the edge segment lengths will have mean value of 1 and a variance of zero, whereas the gap segment lengths will have a mean value equal to the $\lfloor W/2 \rfloor$, where W is the window operator neighborhood size. At low levels of edge gradient threshold the edge detector responses are thick regions and the edge segment length values may vary from 1 to W . The segment length and gap length statistics capture this aspect. Given the true ground truth segment, the edge segment length and gap length statistics and a value for the probability of misdetection of the edge operator, we can generate a realization of the edge detector response by following the procedure outlined in the appendix.

6 Plots and Discussion

The results obtained from the experiments are given in figures 1 through 6. The curves were obtained by taking the running mean of adjacent samples. The window size for the running mean operation was 5. The results shown in the plots are the results obtained after 10 replications. Figures 1 and 4 illustrate how the mean length of the run of edge pixels varies with edge

strength threshold for the morphological operator and the gradient based operator. It is clear from the plots that as the edge strength threshold is increased the run length drops to a value of 1. When the gradient threshold is high, we label lesser number of pixels as edge pixels in the output and hence the runs encountered are of small width. Another point that the plots illustrate is that as the signal to noise ratio increases from -5 to +20 dB the slope of the curve increases. This effect is due to the fact that the noise has the effect of smoothing on the ideal run-length profile. Ideally, we expect the run-length to be a linear function of the threshold (since the input consists of linear ramp edges). Figures 2 and 5 illustrate how the mean gap length varies with the edge strength threshold. As expected, the mean gap length monotonically increases as a function of the edge strength threshold. In the ideal case, we expect the mean gap length to be a linear function of the edge strength threshold and in the presence of large degree of noise this ideal function is blurred. Figures 3 and 6 illustrate how the mean edge pixel positional error varies with edge strength threshold. It is clear that the error drops to zero when the signal to noise ratio is high. When the signal to noise ratio is 0 or 5 dB it can be seen that the mean error is as much as 0.5 pixels. A comparison of the plots for the morphological and gradient based operators indicate that the gradient based scheme is superior for signal to noise levels of 0 dB and higher. The gradient based scheme has comparable errors when the signal to noise ratio is -5 dB. The conclusion in [2] was that the morphological operator had superior false alarm vs misdetect characteristics. The experiments here point out that the morphological operator has poorer localization performance. In a subsequent paper we will attempt to compare the empirical results obtained with theoretical results by utilizing the theoretical expressions for the mean pixel positioning error. The exact expression for the distribution of pixel error for the morphological edge operator discussed in [3] still needs to be worked out. We are also in the process of evaluating the noisy edge generation procedure, that utilizes similar statistics as in our experiments, to see how closely it models errors that occur in real images.

Mean edge run length vs Edge strength threshold
(Morphological Operator)

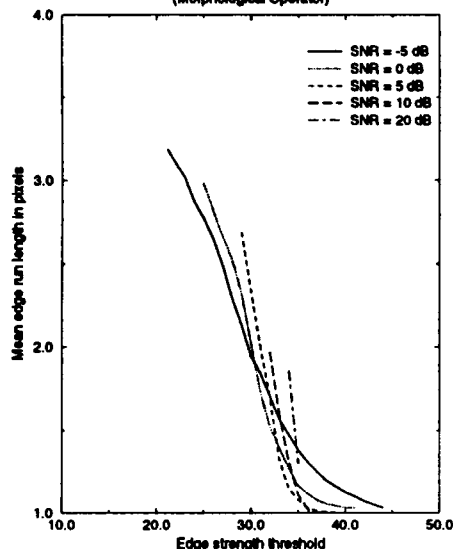


Figure 1: Plot of Mean edge run length vs Edge strength threshold for various signal to noise ratios. Orientation of the true edge was 15 degrees, Window size 5 by 5 for Morphological operator

Edge pixel localization error vs Edge strength threshold
(Morphological operator)

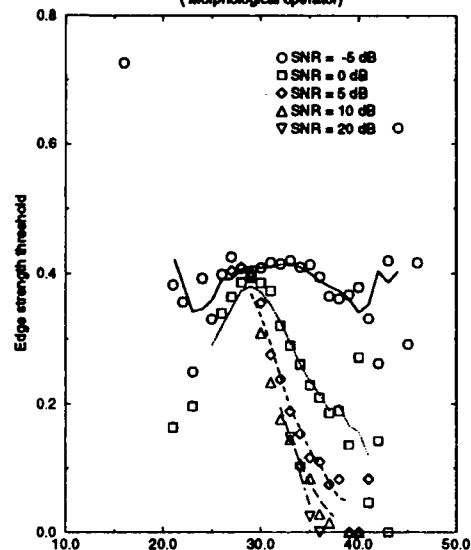


Figure 3: Plot of Mean pixel positional error vs Edge strength threshold for various signal to noise ratios. Orientation of the true edge was 15 degrees, Window size 5 by 5 for Morphological operator

Mean gap length vs Edge strength threshold
(Morphological operator)

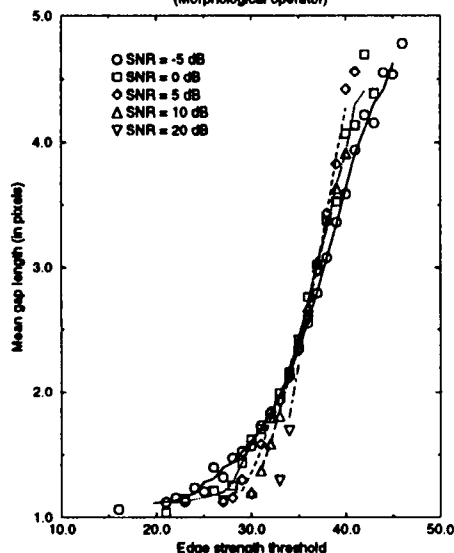


Figure 2: Plot of Mean gap run length vs Edge strength threshold for various signal to noise ratios. Orientation of the true edge was 15 degrees, Window size 5 by 5 for Morphological operator

Mean edge run length vs Edge strength threshold
(Gradient based operator)

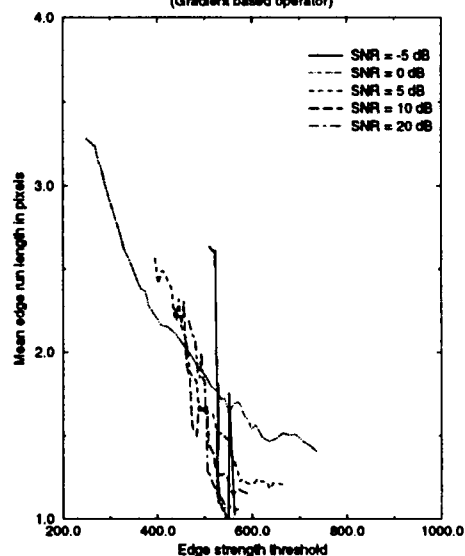


Figure 4: Plot of Mean edge run length vs Edge strength threshold for various signal to noise ratios. Orientation of the true edge was 15 degrees, Window size 5 by 5 for Gradient operator

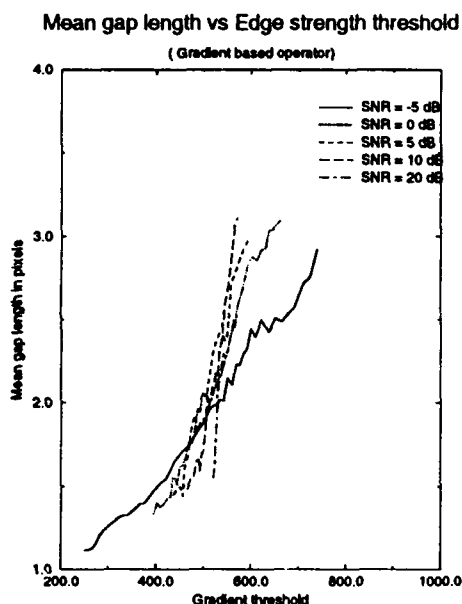


Figure 5: Plot of Mean gap run length vs Edge strength threshold for various signal to noise ratios. Orientation of the true edge was 15 degrees, Window size 5 by 5 for Gradient operator

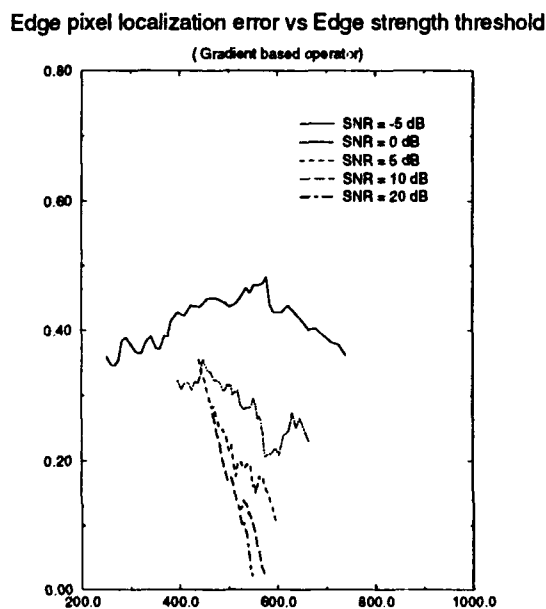


Figure 6: Plot of Mean pixel positional error vs Edge strength threshold for various signal to noise ratios. Orientation of the true edge was 15 degrees, Window size 5 by 5 for Gradient operator

7 Conclusion

In this paper we provided extensions of results provided in [1] and illustrated how one can relax the independence assumptions to derive random perturbation models that include the effects of correlation between neighboring gradient estimates. Under the assumption that the ideal data is corrupted with additive, independent additive gaussian noise, we derived expressions that describe the relationship between an edge gradient estimate at a given location and an edge gradient estimate for a neighboring pixel. We illustrated how the model line breakage process can be modeled as a Markov process whose parameters are functions of the true edge gradient, edge operator's neighborhood size, and the noise variance. Furthermore, we derived theoretical expressions for the mean positional error as a function of the neighborhood operator window size, noise variance, the width of the true ramp edge, and the true edge gradient. We also outlined an experimental protocol used for evaluating edge pixel positioning errors. The results from the experiments illustrate that gradient based edge schemes are superior (when edge localization is of interest) to the morphological scheme discussed in [3]. We also provided an algorithm to generate synthetic noisy edge images that utilize the statistics used in the experiments.

References

- [1] V.Ramesh, R.M.Haralick, "Random Perturbation Models and Performance Characterization in Computer Vision," Proceedings of the CVPR conference, June 1992.
- [2] V.Ramesh, R.M.Haralick, "Performance evaluation of edge operators," Special Session on Performance Evaluation of Modern Edge operators, Orlando Machine Vision and Robotics Conference, 20-24 April 1992.
- [3] J.S.Lee, R.M.Haralick, and L.G.Shapiro, "Morphologic Edge Detection," IEEE Journal of Robotics and Automation, Vol. RA-3, No.2, April 1987.

8 Appendix 1

We assume that the gray values in each neighborhood in the input image can be approximated by computing a planar fit. We assume further that each pixel in the input image is corrupted with additive Gaussian noise, $\eta(R, C)$, with zero mean and variance σ^2 . Let $\hat{\alpha}_{R,C}$, $\hat{\beta}_{R,C}$, and $\hat{\gamma}_{R,C}$ denote the estimates for the coefficients best fitting plane that approximates the N by M neighborhood surrounding the center pixel specified by row and column coordinates (R, C) . In this appendix we derive expressions describing the relationship between the estimates $\hat{\alpha}_{R+i,C+k}$ and $\hat{\alpha}_{R,C}$ for $\alpha_{R+i,C+k}$ and $\alpha_{R,C}$. Let α , β , and γ be the true plane coefficients. It can be easily shown that $\hat{\alpha}_{R,C}$ and $\hat{\beta}_{R,C}$ are equal to:

$$\hat{\alpha}_{R,C} = \alpha + \frac{\sum_{r=-N}^N \sum_{c=-M}^M r \eta(R, C)}{\sum_{r=-N}^N \sum_{c=-M}^M r^2} \quad (9)$$

$$\hat{\beta}_{R,C} = \beta + \frac{\sum_{r=-N}^N \sum_{c=-M}^M c \eta(R, C)}{\sum_{r=-N}^N \sum_{c=-M}^M c^2} \quad (10)$$

Now, we have: $\hat{\alpha}_{R+k,C+j} = \alpha +$

$$\frac{\sum_{r=-N}^N \sum_{c=-M}^M r \eta(R+r+k, C+c+j)}{\sum_{r=-N}^N \sum_{c=-M}^M r^2} \quad (11)$$

The difference, $\hat{\alpha}_{R+k,C} - \hat{\alpha}_{R,C}$ is given by:

$$\begin{aligned} & \sum_{C'=C-M}^{C+M} \sum_{R'=R-N}^{R-N+k-1} (R-R') \eta(R', C') \quad (12) \\ & + \sum_{C'=C-M}^{C+M} \sum_{R'=R+N+1}^{R+N+k} \eta(R', C') \\ & - \sum_{C'=C-M}^{C+M} \sum_{R'=R-N+k}^{R+N} \eta(R', C') \end{aligned}$$

If we visualize the two overlapping neighborhoods, the first term in the above equation corresponds to the difference contributed by the nonoverlapping portion of the left mask, the second term corresponds to the contribution from the nonoverlapping portion of the right mask and the third term corresponds to the contribution from the overlapped portion of the two masks.

Similarly the difference between the $\hat{\beta}$'s, $\hat{\beta}_{R+k,C} - \hat{\beta}_{R,C}$, can be shown to be the ratio of:

$$\sum_{R'=R+N+1}^{R+N+K} \sum_{C'=C-M}^{C+M} c \eta(R', C') - \quad (13)$$

$$\sum_{R'=R-N}^{R-N+K+1} \sum_{C'=C-M}^{C+M} c \eta(R', C')$$

and $\sum_{r=-N}^N \sum_{c=-M}^M c^2$.

The above results can be summarized in a compact fashion using matrix notation. Let $V_{\hat{\alpha}}$ denote a vector consisting of all the estimated $\hat{\alpha}$'s. Let A denote the matrix whose rows contain the values of the kernel used to estimate $\hat{\alpha}$. Let E_{η} denote the vector consisting of the additive Gaussian random variables $\eta(R, C)$'s. It can easily be seen that $V_{\hat{\alpha}} = ME_{\eta} + \alpha 1$. For example, the matrix A for $\begin{pmatrix} \hat{\alpha}_{R,C} \\ \hat{\alpha}_{R+1,C} \end{pmatrix}$ is given by the transpose of the following matrix:

$$\begin{pmatrix} -N & 0 \\ -N+1 & -N \\ -1 & -N+1 \\ \dots & \dots \\ 0 & 1 \\ 0 & -1 \\ 1 & 0 \\ \dots & \dots \\ N & N-1 \\ 0 & N \end{pmatrix} S \quad (14)$$

where $S = 1 / \sum_{r=-N}^N \sum_{c=-M}^M r^2$, and the vector of η 's are given by:

$$\begin{pmatrix} \eta_{R-N,C} \\ \dots \\ \eta_{R+N+1,C} \end{pmatrix} \quad (15)$$

If the η 's are Gaussian random variables with zero mean and covariance matrix Σ then we can see that the values in the vector $V_{\hat{\alpha}}$ is distributed as a multivariate Gaussian random variable with mean $\alpha 1$ and covariance matrix $A \Sigma A'$.

9 Appendix 2 – Procedure for generation of Noisy Edge images

The procedure for generation of noisy edge response images is explained below. First a ground truth edge image is created. This image is now perturbed to obtain the noisy edge image. Let, $G(r,c)$ denote the pixel values in the ground truth ideal edge image, $WSIZE$ denote the edge operator width

PMIS denote the probability of misdetection
 Gap-mean denote the mean gap length
 Gap-variance denote the gap length variance
 Edge-run-mean denote the mean edge run length
 Edge-run-variance denote the edge run length variance
 Then:

```

for each pixel (R,C) in groundtruth image
  if ( G(R,C) == EDGELABEL )
  {
    /* Ground truth pixel is an
       edge pixel */

    X = Uniform(); /* Generate a uniform
                    random number between
                    0 and 1 */
    if ( X < Misdetect-Probability )
    {
      /* Edge pixel is to be perturbed
         according to the edge run and
         gap statistics */

      /* Currently, there are two modes of
         noisy edge generation. Mode 1
         corresponds to the generation of
         gaps and edge runs in the
         direction of edge gradient when
         the user provides the mean and
         variance values for the edge run
         lengths and gap lengths.
      */
      if ( MODE1 )
      {
        /* Generate a gap that is of size
           less the width of the edge
           operator. The gap has to satisfy
           this constraint because the edge
           pixel is deemed to lie within the
           edge zone line (oriented along
           the edge gradient direction and
           is of width WSIZE pixels.) Gsample
           is a procedure that generates a
           sample from a Gaussian distribution.
        */

        AX[0] = Gsample(Gap-mean,
                        Gap-variance);
        while ( X > WSIZE )
          AX[0] = Gsample(Gap-mean,
                          Gap-variance);

        AY[0] = Gsample(Edge-run-mean,
                        Edge-run-variance);
        CURPOS = AX[0] + AY[0];

```

```

    if ( CURPOS > WSIZE )
    {
      /* Output data consists of a gap
         followed by a truncated edge run; */
    }
    else
    {
      i = 1;
      while ( CURPOS < WSIZE )
      {
        AX[i] = Gsample(Gap-mean,
                        Gap-variance);
        AY[i] = Gsample(Edge-run-mean,
                        Edge-run-variance);
        CURPOS += (AX[i] + AY[i]);
        i++;
      }
      /* Output data consists of
         alternating gaps followed
         by runs */
    }
  }

  /* MODE2 corresponds to the
     generation of an edge pixel run
     from the specification of the edge
     positional errors.
  */
  if ( MODE2 )
  {
    X = Gsample(Edge-run-mean,
                Edge-run-variance);
    Y = Gsample(pixel-error-mean,
                pixel-error-variance);
    /* Output data now consists of a
       run of length X centered around
       (R,C) + (r',c'). The values r',c'
       are the coordinates of the pixel
       that is of distance Y from (R,C)
       along the line (oriented along
       the gradient direction).
    */
  }
  else { /* Edge pixel was missed */ }
} /* End of "for each pixel
   in ground truth image" */

```


Section XVIII

Shape from X

Shape from Shadows under Error

David Yang and John R. Kender*

0 Abstract

Kender [Kender and Smith, 1987] presented a method for obtaining shape information from shadows cast at multiple angles of illumination. While the method is correct for perfect data, it may not converge to a solution as a result of noise in the data, digitization of the image, or thresholding errors when determining the shadows. This work extends this shape from shadows algorithm to work in the presence of these factors. The constraints in the original algorithm use trigonometric rules to relate the heights of discrete points (i.e., pixels) based on whether or not the points are in shadow. We supplement these constraints with rules on how shadows in one image constrain those in other images of the same scene and with the same viewpoint, but with different illumination angles. A heuristic is devised to filter the data to satisfy the constraints and produce a solution, and some results are presented for errorful synthetic and real images.

1 Introduction

While shadows hide information in parts of an image (e.g., see [Beckmann, 1965]), researchers have also taken advantage of the relationship between a shadow and the region of the image casting the shadow. The various aspects of this relationship have been used for boundary segmentation [Hambrick and Loew, 1985], spline estimation of surface shape [Hatzitheodorou, 1990], structure hypothesis verification [Irvin and McKeeown, 1989], bounding of a discrete surface profile [Kender and Smith, 1987], surface roughness measurements [Maerz *et al.*, 1990], structure height estimation [Paine, 1981], scene registration and stereo matching [Perlant and McKeeown, 1990], surface gradient measurement under

limited conditions [Shafer, 1985], object boundary determination [Thompson *et al.*, 1987], line drawing interpretation [Waltz, 1975], and microscopic object size measurement [Williams and Wyckoff, 1944].

Unlike intensity-based algorithms like shape from shading, shadow-based algorithms do not require knowledge of the surface reflectance properties of the surface of interest. Implicit in this is the ability to handle surfaces with non-uniform reflectance properties, though multi-colored surfaces and highly specular surfaces may require adjustments in the thresholding process. Below, we give more details on how shadows have been used to determine object heights and surface profiles.

1.1 Shape from shadows

Given the direction of the sun and a reference surface a known distance from the camera, the height of structures on this surface can be determined by basic trigonometry [Paine, 1981]. Specifically, the height of a structure casting a shadow equals the length of the shadow multiplied by $\cotan(\theta_i)$, where θ_i is the incident angle of illumination.

Kender [Kender and Smith, 1987] use multiple lighting positions and trigonometric rules to generate a profile of a surface based on self-shadowing. Their algorithm generalizes the above rule to handle the possibility of self-shadowing and enhances the algorithm with 3 additional constraints. The extra constraints permit the estimation of heights over the entire surface, rather than just the few peak points which can be estimated with the single rule. The only restrictions placed on the surface is that it can be described by a function $z(x, y)$, and is on a reference surface. The first requirement supports a simple model of the surface, while the

*This work was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165

latter requirement allows the accurate determination of the incident angles of illumination.

Since this paper will use much of the terminology of the previous work, we will summarize it here. As can be seen in one possible set-up in fig. 1, the surface of interest is placed on a reference surface, and lighting is placed at several angles in the xz plane and the yz plane. Ideally, the lighting should be collimated and far enough from the surface to approximate lighting from an infinite distance. Intuitively, the lighting in the xz plane is similar to discrete positions of the sun at the equator; thus, east refers to the positive direction along the x -axis, and north, south, and west refer to their respective directions. This also inspired the term *suntrace*, which is the bit matrix (or vector if only a single row or column of the image is being reconstructed) for a given image with a specified illuminant position indicating whether each pixel is shadowed or unshadowed. Equivalently, a suntrace is the overhead view of the thresholded image. Fig. 2b shows the suntrace corresponding to the situation in fig. 2a.

Fig. 2a shows a sample one-dimensional curve with illumination from the west. Since x_2 is in shadow, and x_1 is the closest unshadowed point to the west, x_1 is said to shadow x_2 . Furthermore, if x_2 is not shadowed for any higher western illumination, x_1 is called the *last shadower* of x_2 . From here on, the last shadower will just be called the shadower. Since x_1 is also the closest unshadowed point to the west of the unshadowed x_3 , if x_3 is shadowed for all lower western illuminations, x_1 is called the *failing shadower* of x_3 . For the given shadow region between x_1 and x_3 , x_2 is called the *last shadowed point*, while x_3 is called the *first unshadowed point*. x_0 marks the western boundary of the image, while x_4 marks the eastern boundary.

The 4 constraints can be classified as upper or lower, depending on whether they relate to upper or lower bounds on the height. A forward-upper constraint uses the bound on a shadower to constrain the bound on a shadowed point. A forward-lower constraint does likewise for a failing shadower and a point it fails to shadow. Backward constraints work in the opposite direction. The constraints are as follows:

forward - upper

$$u(x) \leq u(ls(x)) - |ls(x) - x| * sls(x)$$

forward - lower

$$l(x) \geq l(fs(x)) - |fs(x) - x| * sfs(x)$$

backward - upper

$$u(fs(x)) \leq u(x) + |fs(x) - x| * sfs(x)$$

backward - lower

$$l(ls(x)) \geq l(x) + |ls(x) - x| * sls(x)$$

where

$u(x)$ = current upper bound on x

$l(x)$ = current lower bound on x

$ls(x)$ = shadower of x

$fs(x)$ = failing shadower of x

$sls(x)$ = slope of illumination when x is last shadowed

$sfs(x)$ = slope of illumination when x is first unshadowed

Fig. 3 shows synthesized suntrace data for a simple curve and the resulting bounds on the shape.

Hatzitheodorou [Hatzitheodorou, 1990] worked on the related problem of finding the continuous function $z'(x, y)$ which satisfies the forward upper constraints given by the shadow data and minimizes the possible error. His approach was to approximate the actual surface in a shadow region with a spline function. Noise is handled by considering a further constraint on the shape of the solution— e.g., a smoothness constraint.

1.2 Outline for rest of paper

While the shape from shadows algorithm always converges and produces correct bounds on the surface shape when the data is perfect, it may not converge when there are errors in the suntraces. As will be shown, even an error of a single pixel is enough to prevent convergence in the unmodified algorithm. The next section explains the problems faced by the algorithm when dealing with errors in the data. Section 3 describes the heuristic. Section 4 presents results using both synthetic and real imagery. The final section summarizes and suggests future work.

2 Non-convergence

The relaxation algorithm described in [Kender and Smith, 1987] propagates constraints until

no further changes in any bounds are possible. Inconsistencies between 2 or more images, however, may result in a cycle of constraints with a point indirectly constraining its own upper bound to be lower, or its lower bound to be higher. To be concise, from here on, only cycles which indicate such a multiple-point inconsistency will be called cycles. Below, we give an example cycle and then cover two special cases.

2.1 Example cycle

Fig. 4c shows a simple cycle with three points extracted from the suntrace data in Fig. 4a-b. For purposes of the example, assume the upper bound on $x_1 = 3$ is initially 0, and the upper bound on all other points is $+\infty$. The first constraint in the cycle shows $x_1 = 3$ shadowing $x_2 = 25$ when $\theta_i = \arccot(1/3)$. Since x_1 has an upper bound of 0, the forward constraint yields a new upper bound on x_2 of $-7\frac{1}{3}$. The second constraint is another forward constraint, this time with the light on the eastern horizon, producing an upper bound on $x_3 = 13$ of $-7\frac{1}{3}$. The last constraint is backward-upper since $x_4 = x_1 = 13$ is the failing shadower for x_3 when $\theta_i = \arccot(2/3)$ and the illumination is from the west. The result is a new upper bound on $x_4 = x_1$ of $-\frac{2}{3}$. This will prevent termination of the algorithm unless adjustments to the data are made. We call $-\frac{2}{3}$ the *error* in the cycle.

2.2 Mutual shadowers

If we are trying to reconstruct a curve, two points cannot shadow each other (for different light positions) unless they each shadow the other when the light is on one of the opposing horizons. Otherwise, the forward-upper or backward-lower constraints would form a cycle. Intuitively, such an error would imply that one of the points was higher than the other while the other was at least as high. Equivalently, this constraint implies mutual shadowing is valid only when the two points are of the same height. This is reflected in the equations for the forward-upper and backward-lower constraints.

In fig. 3, the mutual shadowing of points $x_1 = 0$ and $x_2 = 3$ can be seen in the suntraces for $\theta_i = \pi/2$ for western and eastern illuminations. Since both suntraces are for illumination at the horizon, there is no error. The reconstruction

indeed estimates both points to be of the same height—note that upper and lower bounds converge at both points. Thus, when valid, mutual shadowing is useful in tightening the bounds.

2.3 Single-point inconsistencies

If x_1 is unshadowed with the light on a given side of the surface—e.g., the right side—then for any higher light position on the same side, x_1 cannot become shadowed again. Fig. 4a shows a case with the western suntraces for $\theta_i = \arccot(2/3)$ and $\theta_i = \arccot(1)$ at $x_1 = 15$. By summing the effect of the forward-upper constraint when x_1 is shadowed with the effect of the backward-upper constraint when x_1 is unshadowed, it can be seen that the upper bound of x_1 and its shadower ($x_2 = 14$) will be lowered indefinitely. Note that in a different surface model, x_1 could legitimately become shadowed again. To see why, consider standing up a slice of swiss cheese on its end and moving a light source vertically up past it on one side. The holes in the cheese will result in unshadowed regions within the shadow cast by the cheese. As the light moves up, these unshadowed regions will become shadowed.

3 Adjustment Heuristic

The chosen adjustment heuristic localizes the blame for an error. It starts by correcting mutual shadowing and single point inconsistent shadowing errors. A local maximum is chosen as the starting point, and both the upper and lower bounds for this point are set to 0. From this point, constraint propagation as in [Kender and Smith, 1987] is performed. Each time a constraint changes a bound on a point, a check for a cycle in the constraints is done. If a cycle is detected, then a point in the cycle is chosen, and the shadowing data for that point is adjusted to either fix the cycle or at least alleviate the problem. The algorithm is explained in detail for processing data for a single vertical slice, and considerations for handling the whole surface are covered as well.

3.1 Initial adjustments

It is noted that when the light is incident from a given direction, all pixels along that boundary of the image should be unshadowed. If not,

this may indicate that these pixels are shadowed by an off-screen object, which could produce errors or overly loose bounds in the results. We assume no such shadowing occurs and mark all boundary pixels on a given side of the image as unshadowed for all suntraces with illumination from that direction.

The next step fixes single-point inconsistencies. Here again there is a problem that if an unshadowed point becomes shadowed again, it is impossible to determine exactly when the point should first be unshadowed in the absence of further information. Favoring the unshadowed status will tend to create reconstructions with spikes. Favoring the shadowed status will tend to create holes in the reconstruction. To avoid both, it may be necessary to first use a low-pass filter on points involved in single-point inconsistencies. Currently, we favor the unshadowed status and do no filtering.

For mutual shadowing, it is usually impossible to know which pixel is wrong. If only one of the points is on the image boundary, it can be reasonably assumed that the other point is wrong, though it is possible that there should be a shadower of one of these points in between them. Otherwise, our current approach arbitrarily favors the point closer to the image boundary for the direction in which it is unshadowed. This point is then marked as unshadowed for the suntraces in which it was originally marked as shadowed by the other point. Note that this does not risk creation of a single-point inconsistency since it marks a point as unshadowed in all suntraces with the light to one side of the surface.

If lighting from both east-west and north-south directions is used, mutual shadowing can involve many more than two points and constraints, but this can become very complicated to handle as a special case. We leave such cases to the general processing of multiple-point inconsistencies.

3.2 Adjusting a shadow edge

We first note that for the 1-dimensional case of a single vertical slice, the only points that need to be considered are those points which are shadowers, last shadowed points or first unshadowed points in at least one suntrace. As a result of the assumption that no points are shadowed by

off-screen objects, this includes all points along any image boundary in the direction of one of the light positions used. The bounds for all other points can be interpolated from the bounds for these points. This can produce a substantial speedup for relatively smooth surfaces (i.e., with few shadowers) and few illuminant positions (i.e., with few last shadowed or first unshadowed points).

An adjustment for a forward-upper constraint can be made by shortening the length of the shadow (starting at the end away from the shadower), thus raising the upper bound of the newly unshadowed points and, when constraints are propagated, of all the other points in the cycle. Similarly, shortening the shadow involved in a backward-lower constraint will decrease the lower bound of the failing shadower. On the other hand, adjustments for either forward-lower or backward-upper constraints require lengthening the shadow.

The amount by which to shorten or lengthen a shadow depends on the total amount by which the cycle lowers the upper bound (or raises the lower bound) of each point in the cycle, the light position for the given constraint, adjacent light positions from the same direction, and possibly the light position of the following constraint in the cycle. For example, consider a point x which is constrained by a forward-upper constraint from the east, and x constrains the next point in the cycle with a forward-upper constraint from the north. Shrinking the shadow for the eastern suntrace will leave x shadowed at the next lower light position. If x 's shadower before the adjustment is still x 's shadower, the effect of the adjustment on the error in the cycle is based on the difference between the slopes of both light positions. On the other hand, if x constrains the next point in the cycle with a backward-upper constraint to the west, the effect of shrinking the shadow for the eastern suntrace can easily be greater than in the previous case. Nonzero slopes of the failing shadows to the west for the points which are newly marked as unshadowed will raise the upper bound of the next point in the cycle even more.

Since each constraint is associated with a particular light position, any adjustment to the data

for this light position should be made compatible with data for other light positions. If a shadow is to be shrunk, the newly unshadowed points must not be shadowed for any higher light position in the same direction to avoid creating a single-point inconsistency. If a shadow is to be expanded, a corresponding check must be done with respect to lower light positions.

3.3 Choosing edge to adjust

To minimize the amount of shadow data adjusted, points involved in constraints of steeper slopes are preferred since small adjustments produce bigger changes in the bounds. It is assumed that the position of shadower points is more likely to be correct than the position of the far shadow boundary. Consequently, adjustments are not made to shadower points after the mutual shadowing errors are fixed. This implies an adjustment will not create a shadower.

3.4 From one dimension to two

Once both north-south and east-west illuminations are used, it is possible to determine a true 2-D map of $z(x, y)$. The 4 trigonometric constraints must now be applied in the 2 extra directions along the new axis. While the extension of the basic height estimation algorithm to 2 dimensions is trivial, the interaction of these constraints does complicate the heuristic. Specifically, when constraints along two axes are used, cycles may have to include points which do not belong to the set of shadowers, last shadowed points, or first unshadowed points. This can happen when consecutive constraints in a cycle operate along different axes. It is still desirable not to adjust (or create) shadowers, so when a point in the middle of the shadow is in a cycle, it is necessary to find the last shadowed point of the shadow at which to make the adjustment.

4 Experimental results

For the real images, we used a set-up similar to that in fig. 1, using uncollimated illumination. We have successfully used the heuristic on both synthetic and real errorful data for 1 dimension, and for synthetic errorful data for 2 dimensions. For the synthetic images, ideal suntraces were derived from a discrete curve or surface.

Errors were then introduced into the suntraces, and the corrupted suntraces were used as input to the modified algorithm. As described earlier, fig. 4 shows data for a simple step-function curve where there is a multiple-point inconsistency. By lengthening the shadow region in the western suntrace where $\theta_i = \text{arccot}(2/3)$ by just 1 pixel to the right, the reconstruction in fig. 4d can be obtained. Fig. 5 shows an 8x8 2-dimensional case. Again, the the fix is a very slight change.

Figs. 6- 8 show sets of 1-dimensional reconstructions along the east-west direction for a wooden half-arch, some egg carton-shaped packaging material, and a crumpled sheet of aluminum foil, materials with very different reflectance properties. For clarity, not all row reconstructions are shown. The images of the original objects have illumination incident from the left at $\text{arccot}(1/3)$. The aperture was generally set a little too wide to facilitate thresholding. The angles of incident light used were $\pi/2$, $\text{arccot}(1/3)$, $\text{arccot}(2/3)$, $\text{arccot}(1)$, and $\text{arccot}(10)$.

For the rows tested for the wooden block, there were no multiple-point inconsistencies for some of the rows north and south of the block, and from 2 to 7 such inconsistencies in the other rows. On a lightly loaded Sun 4, the cpu time needed for one row, as measured by the Unix *time* command, was no more than 0.3 seconds. The real world time needed was in all cases no more than 2.3 seconds. For the packaging material, the number of multiple-point inconsistencies detected varied from 31 to 151, though some cycles were found more than once. The cpu time for processing one row varied from 2 to 15 seconds, and the real world time varied from 13 seconds to 1 minute and 22 seconds. For the foil, the number of cycles detected varied from 3 to 76. The cpu time for processing one row varied from 1 to 7 seconds, and the real world time varied from 9 to 42 seconds. The upper bounds are generally more accurate than the lower bounds, especially for certain points on the far left and far right sides of the reconstruction. Points on one side of the image are sometimes in shadow for all light positions from the other side. Such a point will not be constrained by any forward-lower constraint. If the point does not shadow any points, it will not be constrained by any backward-lower constraint, either, in which case

the lower bound will remain at $-\infty$.

It is unsurprising that the packaging material caused the most trouble. The material is spongy, so besides the general sinusoidal shape, there is roughness along the entire surface. This kind of texture, plus the darker color of the material, leads to more thresholding errors. There are also more local maxima which shadow reasonably-sized areas. Backward-upper constraints produce an upper bound on the constrained point that is at least as high as the bound on the constraining point, so all upper bound cycles must include at least one shadower. The same is true for forward-lower constraints and lower bound cycles. In the case of sinusoidal surfaces like the packaging material, the sets of points in the cycles tend not to intersect since the shadowers are not themselves shadowed except for light positions near the horizon. Fixing one cycle is thus unlikely to fix other inconsistencies.

5 Discussion and future work

The most immediate goal is to either prove that the heuristic in fact always converges, or determine a general description of cases where it does not. As it turns out, the decision to choose the point where the greatest effect can be achieved can result in non-convergence. This is possible since an adjustment that raises the upper bound of some points will tend to raise the lower bound as well. The fix for an upper bound cycle may thus create a lower bound cycle. However, we have not experienced this problem in our experiments. If we treat the set of suntraces for the given images as a state within a state space, then the current approach can be seen as a kind of depth-first search in that a next state is chosen by fixing the first cycle found and considering only one of the possible fixes for the cycle. Guaranteeing convergence probably requires adding some breadth to the search.

Ideally, we would like to find a method for quickly determining if a set of suntrace data contains a constraint cycle (i.e., without actually propagating constraints until a cycle is found), and if it does, what is the closest set of suntrace data which does not. The closest set would be chosen using some definition of minimal adjustment to the actual data.

References

- [Beckmann, 1965] P. Beckmann. Shadowing of random rough surfaces. *IEEE Trans. on Antennas and Propagation*, 13:384-8, 1965.
- [Hambrick and Loew, 1985] L. Hambrick and M. Loew. Entry-exit method of shadow boundary segmentation. In *CVPR*, pages 656-8, 1985.
- [Hatzitheodorou, 1990] M. Hatzitheodorou. *Shape From Shadows: Theoretical and Computational Aspects*. PhD thesis, Columbia U., Comp. Sci. Dept., 1990.
- [Irvin and McKeown, 1989] R. Irvin and D. McKeown. Methods for exploiting the relationship between buildings and their shadows in aerial imagery. *IEEE SMC*, 19(6):1564-75, 1989.
- [Kender and Smith, 1987] J. Kender and E. Smith. Shape from darkness: Deriving surface information from dynamic shadows. In *AAAI*, pages 539-46, 1987.
- [Maerz et al., 1990] N. Maerz, J. Franklin, and C. Bennett. Joint roughness measurement using shadow profilometry. *Int'l. J. of Rock Mech., Mineral Sci. & Geomech.*, 27(5):329-43, 1990.
- [Paine, 1981] D. Paine. *Aerial Photography and Image Interpretation for Resource Management*. John Wiley & Sons, New York, 1981.
- [Perlant and McKeown, 1990] F. Perlant and D. McKeown. Scene registration in aerial image analysis. *Photogrammetric Eng. and Remote Sensing*, 56(4):481-93, 1990.
- [Shafer, 1985] S. Shafer. *Shadows and Silhouettes in Computer Vision*. Kluwer, Boston, 1985.
- [Thompson et al., 1987] W. Thompson, M. Checky, and W. Kaemmerer. Shadow stereo - locating object boundaries using shadows. In *AAAI*, pages 761-6, 1987.
- [Waltz, 1975] D. Waltz. Understanding line drawings of scenes with shadows. In P. H. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, New York, 1975.
- [Williams and Wyckoff, 1944] R. Williams and R. Wyckoff. The thickness of electron microscopic objects. *J. of Appl. Physics*, 15:712-6, 1944.

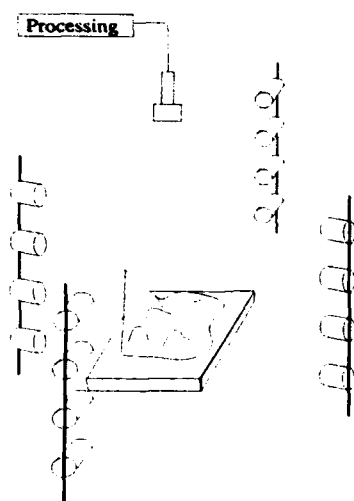


Figure 1: One possible set-up for shape from shadows

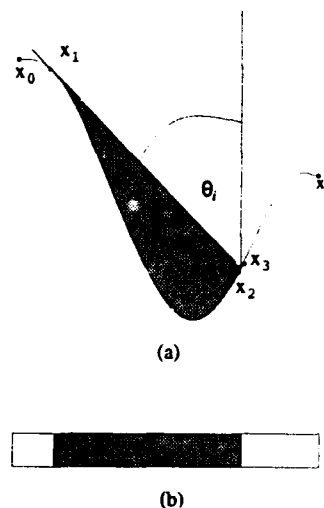


Figure 2: (a) A sample curve – the shaded region is the shadowed portion for the given angle of illumination (b) The corresponding suntrace

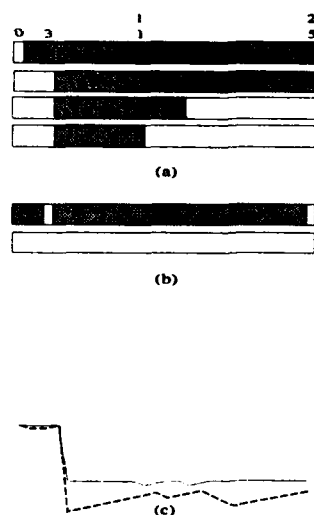


Figure 3: (a) Suntraces for western illuminations – top to bottom, angles of illumination = $\pi/2$, $\text{arccot}(1/3)$, $\text{arccot}(2/3)$, $\text{arccot}(1)$ (b) Suntraces for eastern illuminations – top to bottom, angles of illumination = $\pi/2$, $\text{arccot}(1/3)$ (c) Reconstruction – solid line is upper bound, dashed is lower bound

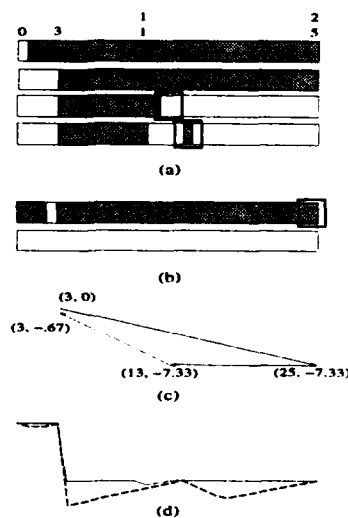


Figure 4: (a) Suntraces for western illuminations – top to bottom, angles of illumination = $\pi/2$, $\text{arccot}(1/3)$, $\text{arccot}(2/3)$, $\text{arccot}(1)$ [Introduced errors are boxed here and in part (b)] (b) Suntraces for eastern illuminations – top to bottom, angles of illumination = $\pi/2$, $\text{arccot}(1/3)$ (c) Constraint cycle revealing error (d) Reconstruction

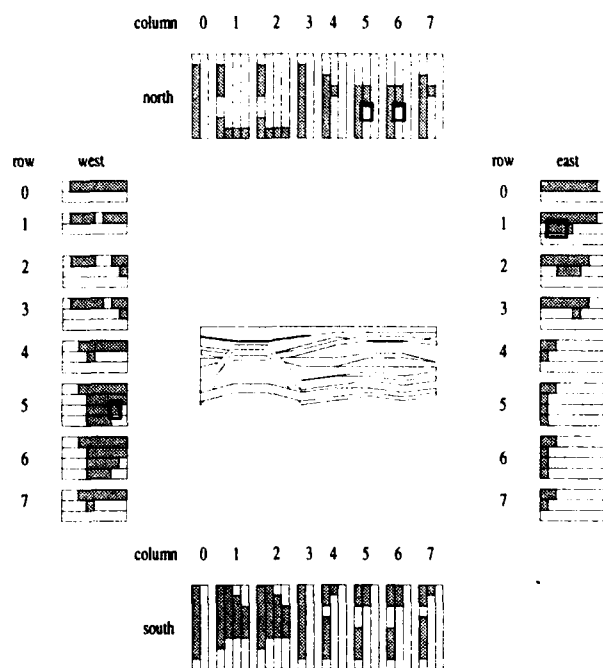


Figure 5: Suntraces and surface reconstruction of errorful synthetic 8x8; angles of illumination = $\pi/2$, $\arccot(1/3)$, and where included, $\arccot(2/3)$, $\arccot(1)$ [Introduced errors are boxed]

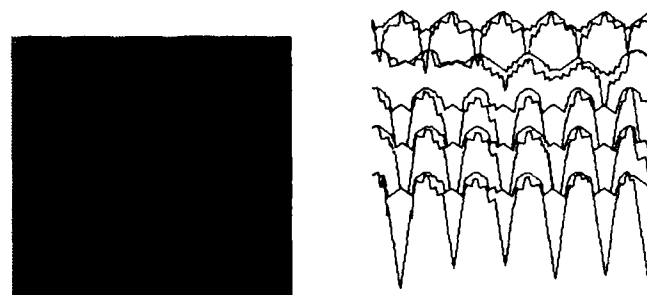


Figure 7: Reconstruction of foam surface profiles

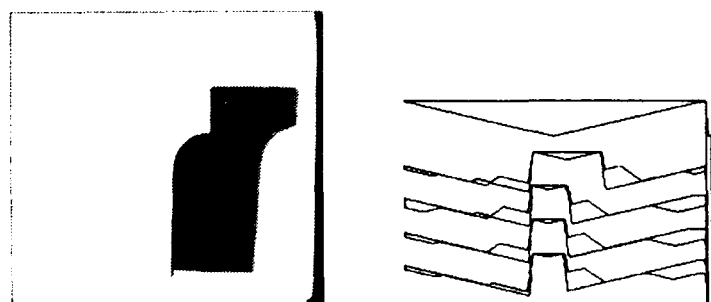


Figure 6: Reconstruction of wood half-arch profiles [Viewing angle for reconstruction is $\pi/4$, as in the next 2 figures]

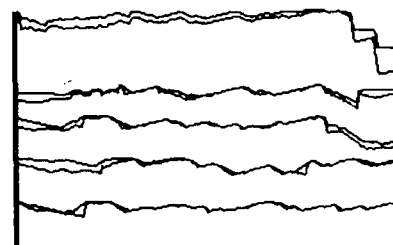


Figure 8: Reconstruction of foil surface profiles

Shape and Motion from Linear Features*

Warren F. Gardner and Daryl T. Lawton

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280

Abstract

This paper introduces a technique for extracting structure and motion using directionally selective matches between linear features. A world-centered coordinate system is used to make these computations without the intermediate calculation of depth. In order to constrain the possible structure and motion configurations, we assume that the three-dimensional direction of gravity relative to each image frame is known. The direction of gravity, along with the directionally selective linear feature matches, form a set of quadratic equations which can be used to determine structure and motion.

1 Introduction

The extraction of environmental structure and motion from a sequence of two-dimensional images is a common problem in computer vision. Typically solutions to this problem are expressed in camera-centered coordinate systems where environmental geometry is specified by the depth along an image feature's ray of projection. Unfortunately, parameters computed from this camera-centered representation are dependent upon the depth to environmental features. This leads to erroneous results for objects located far from the camera.

The recently introduced *factorization method* [Tomasi and Kanade, 1990; Tomasi and Kanade, 1992; Boult and Brown, 1992] has attempted to overcome the disadvantages associated with a camera-centered representation. This method uses a world-centered coordinate system, along with an orthogonal projection assumption, in order to compute shape and motion without the intermediate calculation of depth. A matrix of image measurements is constructed by making point correspondences between image frames. The matrix is then factored into a shape matrix and a motion matrix using Singular Value Decomposition.

One problem with the factorization method is that it relies upon accurate point correspondences between im-

age frames. This paper introduces a method of extracting shape and motion from directionally selective linear feature correspondences. This line-based algorithm is capable of reconstructing shape and motion without computing depth as an intermediate step. In addition to the orthogonality assumption, we assume that the three-dimensional direction of gravity is known relative to each image in a motion sequence.

The algorithm begins by searching for the orientation of one of the lines in the environment. This is a one dimensional search over 180° , constrained by the projection of the line on one of the image planes. Each candidate line orientation, along with the position of gravity, forms a set of quadratic equations which constrain all the other lines, as well as the rotation between image frames. An error measure is computed from the derived line orientations and used to evaluate each shape and motion configuration. Once the line orientations and parameters of rotation have been derived, the relative positions of the lines can also be computed from simple linear equations.

The remainder of this section introduces the notation used throughout this paper. Section 2 shows how to derive line orientation and camera rotation from a sequence of two-dimensional images. Section 3 presents a set of linear equations which can be used to solve for the relative line positions. The algorithms presented in the paper are applied to synthetic data and the results are presented in Section 4. Finally, concluding remarks are given in Section 5.

1.1 Notation

The notation used throughout this paper is shown in Figure 1. An image frame at time f is delineated by unit vectors i_f , j_f , and k_f . A three-dimensional environmental line is represented by a unit vector d , specifying the line direction, and a point on the line p . Line (d, p) is projected orthographically onto image frame f . The direction of the projected line is represented by its unit normal \tilde{n}_f . \tilde{p}_f refers to the projection of p . The direction of gravity will be referred to as g_f . The two-dimensional parameters \tilde{n}_f and \tilde{p}_f , as well as the three-dimensional parameter g_f are all expressed in the coordinate system of image frame f . All other parameters are specified relative to the world coordinate system. When \tilde{n}_f is specified in the world coordinate system it

*This research is supported by the Advanced Research Projects Agency of the Department of Defense and is monitored by the U. S. Army Topographic Engineering Center under contract No. DACA76-92-C-0016

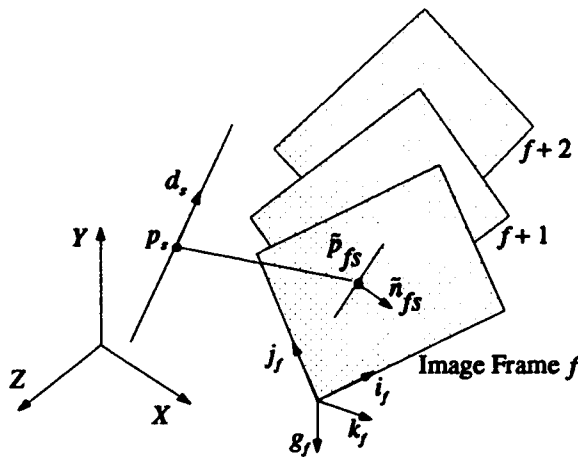


Figure 1: Coordinate systems

will be referred to as n_{fs} .

In the following section we present a method of solving for the line orientations d_s , as well as the parameters of rotation i_f , j_f , and k_f . Section 3 shows how these initial quantities can be used to fix the relative positions of the lines within the world coordinate system by solving for a point p_s on each line.

2 Line Orientation and Camera Rotation

In this section we present a method of solving for the three-dimensional line orientations and parameters of rotation from a sequence of two-dimensional images. The algorithm begins by searching for the orientation of one of the lines. This is a one dimensional search over 180° , constrained by the projection of the line on one of the image planes. Each candidate line orientation, along with the position of gravity, forms a set of quadratic equations which constrain all the other lines, as well as the rotation between image frames. An error measure is computed from the derived line orientations and used to evaluate each shape and motion configuration. Section 2.1 shows how to solve for the position within the world coordinate system of the line normals (n_{fs}) associated with the candidate line. Section 2.2 shows how the candidate normals can be used to solve for the normals to all the other visible lines. These line normals are then used in Section 2.3 to estimate the line orientations and camera rotations.

2.1 Candidate Line Normals

The algorithm begins by searching for the orientation of one of the lines. A candidate line is used to constrain the position of all the other three-dimensional lines so that a particular shape and motion arrangement can be evaluated. The first step in this process is to solve for the position in the world coordinate system of the candidate line's normals. Let d_1 be the candidate line.

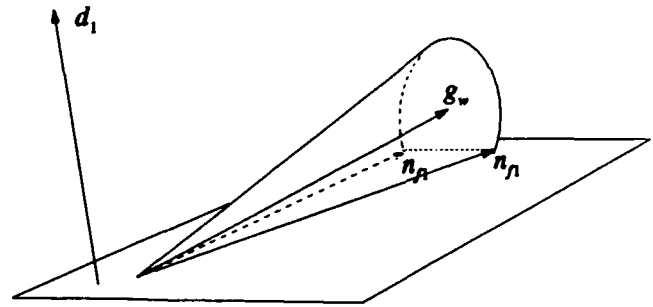


Figure 2: Normals are determined by intersecting a plane with a circular cone

Since the line normals \tilde{n}_{f1} were formed by orthographic projection, they must be perpendicular to the line d_1 . Therefore, one constraint is that the vectors n_{f1} must lie within the plane perpendicular to d_1 . An additional constraint is provided by the gravity vector g_f . The angle between \tilde{n}_{f1} and g_f must be the same as the angle between n_{f1} and the direction of gravity in the world coordinate system (g_w). These two constraints can be used to solve for n_{f1} . Figure 2 shows the geometry of these two constraints. Each normal (n_{f1}) is determined by intersecting a plane with a circular cone. The plane is defined by d_1 . The cone is constructed by rotating a vector about the direction of gravity at the appropriate angle. Since the origin of the cone lies within the plane, the intersection of the plane with the cone results in two lines. There are only two possible solutions since the normals are known to be unit vectors.

The constraints described above will now be examined in more detail. As stated earlier, the direction of gravity g_f relative to the line normals \tilde{n}_{f1} is known. This results in the following relationship

$$n_{f1} \cdot g_w = \tilde{n}_{f1} \cdot g_f \quad (1)$$

where g_w is the direction of gravity in the world coordinate system. Letting $g_w = (0, -1, 0)$ we can simplify Equation 1

$$n_{f1_y} = -\tilde{n}_{f1} \cdot g_f \quad (2)$$

In addition to the angle constraint we know that n_{f1} lies within the plane defined by d_1 . This constraint is expressed as

$$n_{f1} \cdot d_1 = 0 \quad (3)$$

Finally, we know that the magnitude of each normal vector (n_{f1}) equals one

$$\|n_{f1}\| = 1 \quad (4)$$

Equations 2, 3, and 4 can be combined into a single quadratic equation, resulting in two feasible solutions for each normal vector.

2.2 Additional Line Normals

The next step in the extraction of line orientation and rotation is to solve for the position within the world coordinate system of the rest of the line normals. This is

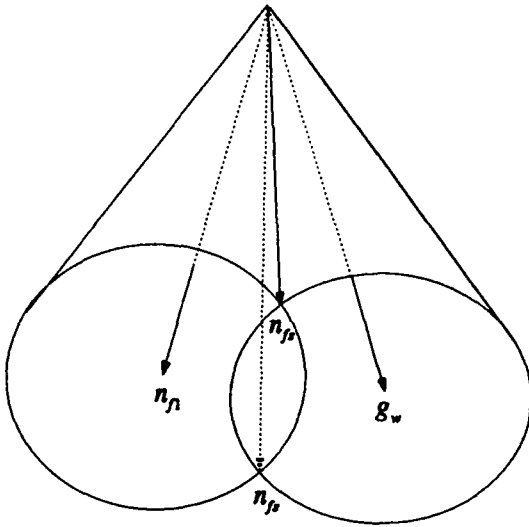


Figure 3: Normals are determined by intersecting two circular cones

accomplished by using the candidate line normals. The idea is essentially the same as in the previous section. Two constraints can be formulated from the given geometry. The first constraint is given by the gravity vector g_f , and is identical to the constraint presented in the previous section. The angle between \tilde{n}_{fs} and g_f must be the same as the angle between n_{fs} and the direction of gravity in the world coordinate system. The second constraint is that the angle between an image normal vector \tilde{n}_{fs} and the candidate image normal vector \tilde{n}_{f1} must be the same as the angle between the associated world coordinate normal vectors n_{fs} and n_{f1} . These two constraints can be used to solve for all the additional normal vectors n_{fs} . The constraints are shown geometrically in Figure 3. The solution for a normal vector n_{fs} is essentially the result of intersecting two circular cones. One cone is the result of rotating a vector about the direction of gravity. The other cone results from rotating a vector about the candidate normal vector n_{f1} . The intersection of two circular cones which share the same origin is two lines. Once again, the normals are known to be unit vectors, resulting in two solutions.

The following equations result from the above analysis. The constraint resulting from the gravity vector g_f is identical to the one presented in Section 2. Therefore, from Equation 2 we can write

$$n_{fs} = -\tilde{n}_{fs} \cdot g_f \quad (5)$$

The second constraint relates the line normals n_{fs} to the candidate line normals n_{f1} as follows

$$n_{fs} \cdot n_{f1} = \tilde{n}_{fs} \cdot \tilde{n}_{f1} \quad (6)$$

Finally, we know that the magnitude of each normal vector (n_{fs}) equals one

$$\|n_{fs}\| = 1 \quad (7)$$

Equations 5, 6, and 7 can be combined into a single quadratic equation, resulting in two feasible solutions for each normal vector.

2.3 Parameter Estimation

Once the normal vectors (n_{fs}) have been derived, the process of estimating the line orientations and rotational parameters is trivial. The line orientations (d_s) are easily estimated from their associated normals (n_{fs}) using the following equation

$$d_s \cdot n_{fs} = 0 \quad (8)$$

d_s can be estimated with a minimum of two non-collinear normal vectors. When more vectors are available, d_s can be solved for using a linear least-squares technique. The rotational parameters are also easily obtained from the normal vectors n_{fs} . Three linear equations can be formulated for the three rotational parameters i_f , j_f , and k_f

$$\begin{aligned} i_f \cdot n_{fs} &= \tilde{n}_{fs_x} \\ j_f \cdot n_{fs} &= \tilde{n}_{fs_y} \\ k_f \cdot n_{fs} &= 0 \end{aligned}$$

There are also additional constraints available. One of these constraints is that the vectors must be orthonormal

$$\begin{aligned} i_f &= j_f \times k_f \\ j_f &= k_f \times i_f \\ k_f &= i_f \times j_f \\ \|i_f\| &= \|j_f\| = \|k_f\| = 1 \end{aligned}$$

Additional constraints can be derived from the relationship between the rotational vectors and gravity as was done in Sections 2.1 and 2.2. These constraints are

$$\begin{aligned} i_f \cdot g_w &= g_{fs_x} \\ j_f \cdot g_w &= g_{fs_y} \\ k_f \cdot g_w &= g_{fs_z} \end{aligned}$$

Of course, all of the equations presented above are not independent, and all are not necessary. Currently we use the following subset of equations. Initially k_f is determined using a least squares formulation of

$$k_f \cdot n_{fs} = 0 \quad (9)$$

The technique presented in Section 2.1 is then used to solve for i_f with the following equations

$$i_f \cdot k_f = 0 \quad (10)$$

$$i_f \cdot g_w = g_{fs_x} \quad (11)$$

Finally i_f and k_f are used to solve for j_f

$$j_f = k_f \times i_f \quad (12)$$

Equation 8 is used to solve for the line orientations d_s . Equations 9, 10, 11, and 12 are used to solve for the rotational parameters i_f , j_f , and k_f . The following section shows how to use these derived parameters to solve for the relative positions of the line segments, thus completing the spatial reconstruction.

3 Line Position

The final step in the line segment reconstruction is to solve for the line segment positions relative to the world

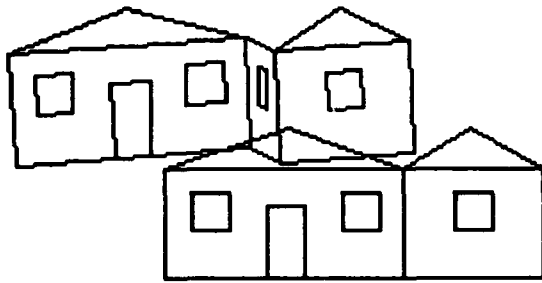


Figure 4: The first and last frames from a 20 image sequence

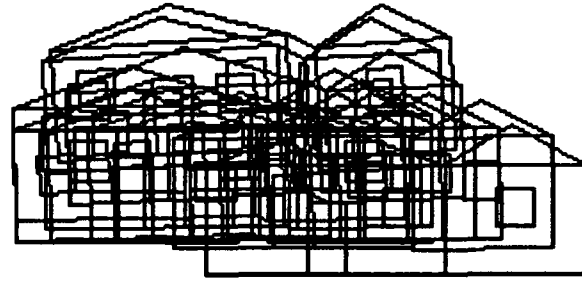


Figure 5: 10 image frames from a 20 image sequence

coordinate system. Initial assumptions about the position of the image frames relative to the world coordinate system are made, allowing a simple linear solution to the problem. The position of each line is represented by a point p_s , which is chosen arbitrarily. The world coordinate system will be positioned at the center of image frame 1. The points \tilde{p}_{1s} are then chosen arbitrarily $\tilde{p}_{1s} = (x_s, y_s)$. We assume that all the image planes intersect along line d_1 . This means that the position of each image plane is given by $p_1 + \alpha_f d_s$, where α_f is a parametric scale factor.

Each line position $p_s = (x_s, y_s, z_s)$ consists of one unknown z_s . The solution for z_s is trivial. Each point p_s is constrained to lie within the planes perpendicular to n_{fs} . These planes are positioned by choosing some arbitrary point on the projection of each line, and then determining the position of that point within the world coordinate system. Let q be the point in world coordinates

$$q = p_1 + [i_f \ j_f] \cdot (\tilde{p}_{fs} - \tilde{p}_{f1}) \quad (13)$$

The equation of the plane is then written as

$$\begin{aligned} n_{fs,x}(x_s - q_x - \alpha_f d_{s,x}) + \\ n_{fs,y}(y_s - q_y - \alpha_f d_{s,y}) + \\ n_{fs,z}(z_s - q_z - \alpha_f d_{s,z}) = 0 \end{aligned} \quad (14)$$

The two unknowns in this equation are z_s and α_f . α_f can be removed from the equation, and a least squares solution can be found for z_s .

4 Results

The algorithm presented in this paper was implemented and tested on several sequences of synthetic data. The first and last frames from a 20 image sequence are shown in Figure 4. Figure 5 shows 10 frames from the sequence (every other frame is displayed). This data was produced by random rotations and translations. The rotational parameters i_f and j_f associated with this sequence of motion are shown in Figures 6 and 7. The correct rotational values are displayed as solid lines, and the derived values are displayed as dotted lines. All errors are the result of perspective projection. Notice that

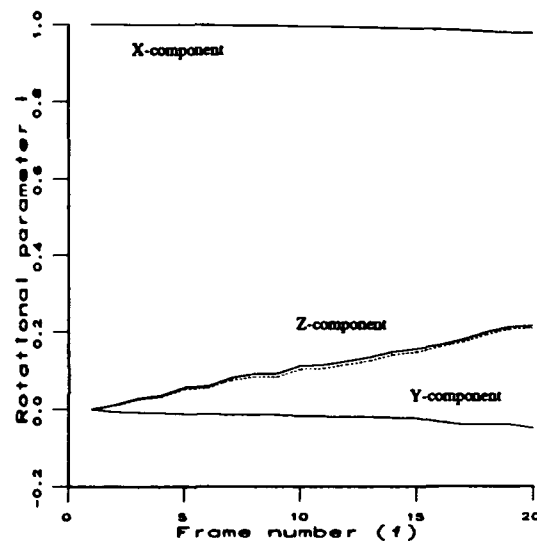


Figure 6: The components of i_f for a 20 frame sequence. The correct values are shown with solid lines, and the derived values are shown with dotted lines.

the Y-component of i_f is errorless. This is because this component is derived from the relationship between the image frames and the gravity vector (g_f) as shown in Equation 11. Thus the Y-component is unaffected by the perspective projection errors.

The derived line orientations and parameters of rotation were then used to reconstruct the line positions as discussed in Section 3. A top view of the original data is shown in Figure 8. The reconstructed data is shown in Figure 9. Once again the errors are the result of perspective projection.

5 Conclusion

The technique presented in this paper is an early attempt at constructing linear feature based depth-independent motion algorithms. The work has only been tested on synthetic data, and it is not clear what effect perspective

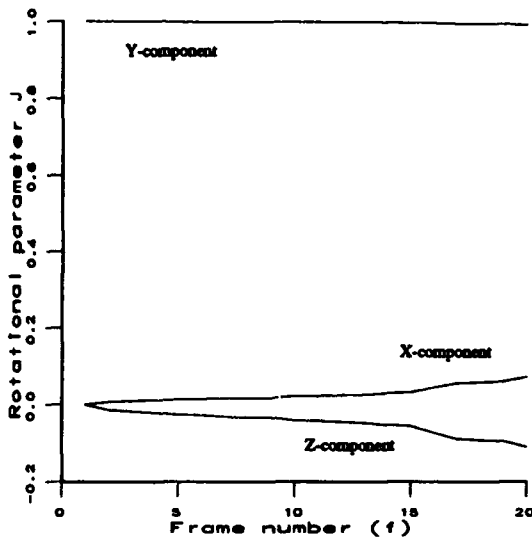


Figure 7: The components of j_f for a 20 frame sequence. The correct values are shown with solid lines, and the derived values are shown with dotted lines.

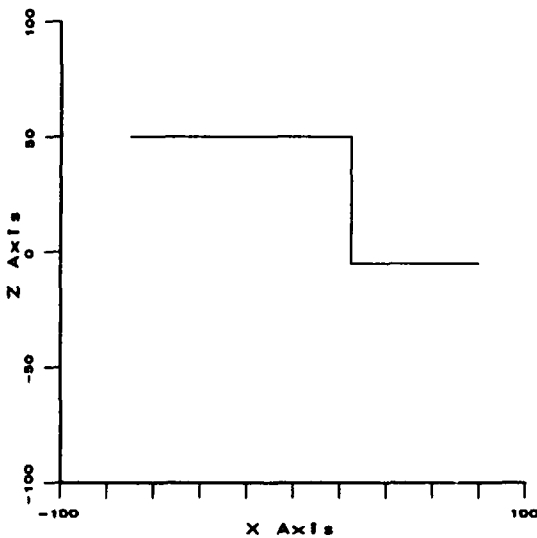


Figure 8: Top view of the house data

projection and other forms of noise will have. However, since the formulation involves linear least squares estimation, it appears that it will be robust. The ability to deal with occlusion is also straight-forward in this over-constrained system. Occluded line normals (n_{fs}) are null vectors and therefore have no effect on the least squares solution. Notice that the first frame shown in Figure 4 contains occluded lines.

One drawback of this method is that the three-dimensional direction of gravity is required. This measurement can be provided by a gravity sensor, but we would like to relax this restriction. One way to remove the gravity vector from the algorithm is to replace the di-

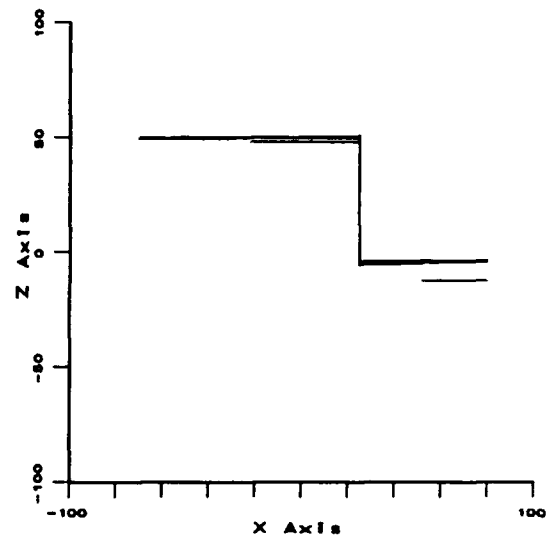


Figure 9: Top view of the reconstructed house

rection of gravity with another consistent direction. For example, for an object that consistently moves in one direction (such as a vehicle), the gravity vector can be replaced by a vector specifying this direction (the forward vehicle direction).

There are several areas for future work:

- Test this algorithm on noisy data and if necessary develop a more robust formulation that will work well in the presence of errors, including the errors introduced from perspective projection.
- Test the algorithm on real image sequences.
- Integrate this rotation based method with the translation based method discussed in [Lawton, 1982]. In this case the gravity vector is replaced by a direction of translation vector. The integration of these two methods will probably be accomplished through temporal filtering using the Kalman Filter.

References

- [Boulton and Brown, 1992] Terrance E. Boulton and Lisa Gottesfeld Brown. Motion segmentation using singular value decomposition. In *Proceedings of the Image Understanding Workshop*, pages 495-506, 1992. San Diego, CA.
- [Lawton, 1982] Daryl T. Lawton. Processing translational motion sequences. *Computer Vision, Graphics, and Image Processing*, 22:116-144, 1982.
- [Tomasi and Kanade, 1990] Carlo Tomasi and Takeo Kanade. Shape and motion without depth. In *Proceedings of the Image Understanding Workshop*, pages 258-270, 1990. Pittsburgh, PA.
- [Tomasi and Kanade, 1992] Carlo Tomasi and Takeo Kanade. The factorization method for the recovery of shape and motion from image streams. In *Proceedings of the Image Understanding Workshop*, pages 459-472, 1992. San Diego, CA.

Object-Centered Surface Reconstruction: Combining Multi-Image Stereo and Shading*

P. Fua and Y. Leclerc
SRI International
333 Ravenswood Avenue, Menlo Park, CA 94025
(fua@ai.sri.com leclerc@ai.sri.com)

Abstract

Our goal is to reconstruct both the shape and reflectance properties of surfaces from multiple images. We argue that an object-centered representation is most appropriate for this purpose because it naturally accommodates multiple sources of data, multiple images (including motion sequences of a rigid object), and self-occlusions. We then present a specific object-centered reconstruction method and its implementation. The method begins with an initial estimate of surface shape (provided by triangulating the result of conventional stereo or other means). The surface shape and reflectance properties are then iteratively adjusted to minimize an objective function that combines information from multiple input images. The objective function is a weighted sum of "stereo," shading, and smoothness components, where the weight varies over the surface. For example, the stereo component is weighted more strongly where the surface projects onto highly textured areas in the images, and less strongly otherwise. Thus, each component has its greatest influence where its accuracy is likely to be greatest. Experimental results on both synthetic and real images are presented.

1 Introduction

The problem of recovering the shape and reflectance properties of a surface from multiple images has received considerable attention [6, 20, 35, 44]. This is a key problem not only in

developing general-purpose vision systems, but also in specialized areas such as the generation of Digital Elevation Models from aerial images [5, 12, 26, 53].

In this paper, we view the recovery problem as one of finding an object-centered description of a surface from a set of input images that is sufficiently complete, in terms of its geometric and radiometric properties, that it is possible to generate an image of the surface from any viewpoint. In particular, the description should be sufficiently complete to reproduce the input images to within a certain tolerance, given models of the cameras, their relative locations, and expected noise.

Our surface reconstruction method uses an object-centered representation, specifically, a triangulated 3-D mesh of vertices. Such a representation accommodates the two classes of information mentioned above, as well as multiple images (including motion sequences of a rigid object) and self-occlusions. We have chosen to model the surface material using the Lambertian reflectance model with variable albedo, though generalizations to specular surfaces would be relatively straightforward. Consequently, the natural choice for the monocular information source is shading, while intensity is the natural choice for the image feature used in multi-image correspondence. Not only are these the natural choices given a Lambertian reflectance model, they are also complementary [7, 30]: intensity correlation is most accurate wherever the input images are highly textured, whereas shading is most accurate when the input images are untextured.

*Support for this research was provided by various contracts from the Defense Advanced Research Projects Agency.

The reconstruction method is to minimize an objective function whose components depend on the input images and some measure of the complexity of the 3-D mesh. The method starts with an initial estimate for the mesh derived from the triangulation of conventional stereo results, and uses a standard optimization technique called *conjugate gradient descent* to minimize the objective function. The image-dependent components of the objective function are related to the two sources of information mentioned above. We take advantage of the complementary nature of the information sources by weighting the components at each facet of the triangulated mesh according to the amount of texturing within the area of the images that the facet projects to. The projection uses a hidden-surface algorithm to take occlusions into account.

In the following section, we describe related work and our contributions in this area. Following this we discuss some of the key issues in multi-image surface reconstruction and how to combine different sources of information for such purposes. We then describe in detail our specific procedure, discuss the behavior of our procedure on synthetic data, and show some results on real images.

2 Related Work and Contributions

Three-dimensional reconstruction of visible surfaces continues to be an important goal of the computer vision research community. Initially, much of the work concentrated on $2\frac{1}{2}$ -dimensional image-centered reconstructions, such as Barrow and Tenenbaum's *Intrinsic Images* [6] and Marr's $2\frac{1}{2}$ -D *Sketch* [35]. These preliminary ideas have been the basis for quite successful systems for recovering shape and surface properties. Some have used single sources of information, such as sequences of range data or intensity images [3, 25], stereo [12, 26, 52, 53], and shading [21, 24, 44]. Others have combined sources of information, such as shading and texture [8], features and stereo [23], focus, vergence, stereo, and camera calibration [1]. See [2] for further discussions on

information fusion.

More recently, full 3-dimensional models have been used, such as 3-D surface meshes [46, 49], parameterized surfaces [40, 33], particle systems [42, 17], and volumetric models [36, 45, 37].

As with the $2\frac{1}{2}$ -dimensional representations, 3-D representations have used a variety of single image cues for reconstruction, such as silhouettes and image features [9, 11, 47, 48, 50], range data [51], stereo [17], and motion [41]. Liedtke[32] first uses silhouettes to derive an initial estimate of the surface, and then uses a multi-image stereo algorithm to improve on the result. Their approach to deriving an initial estimate for the mesh, as with Szeliski and Tonneson's approach [42], is significantly more powerful than the one we use in this paper. This is an important topic for future research.

Of special relevance to this paper is research in combining stereo and shape from shading. Using $2\frac{1}{2}$ -dimensional representations, Blake *et al.* [7] is the earliest reference we are aware of that discusses the complementary nature of stereo and shape from shading, but their experimental results are almost non-existent in this paper. Leclerc and Bobick [31] discuss the integration of stereo and shape from shading, but only use stereo as an initial condition to a discrete height from shading algorithm. Cryer *et al.* [10] combine the high-frequency information from a shape from shading algorithm with the low-frequency information from a stereo algorithm using filters designed to match those in the human visual system.

Using full 3-D representations, Heipke [22] integrates the two cues, but assumes that the images can be separated beforehand into zones of variable albedo (where one does stereo) and areas of constant albedo (where one does shape from shading). This is in contrast to our approach, in which the optimization procedure dynamically adapts to the image data.

In this paper, we unify the idea of using 3-D meshes to integrate information from multiple images with that of using multiple cues. Our specific approach to this unification, has led to a number of important contributions:

- We correctly deal with occlusions by using a hidden surface algorithm during the re-

construction process.

- Our technique for doing stereo avoids the constant depth assumption of traditional correlation-based stereo algorithms, effectively using variable-sized windows in the images.
- Our approach to shape from shading is applicable to surfaces with slowly varying albedo. This is a significant advance over traditional approaches that require constant albedo.
- We propose a dynamic weighting scheme for combining shape from shading and stereo, and demonstrate that it leads to significantly better results than using either cue alone using both synthetic and real images.

To demonstrate the validity of the overall approach, we have implemented a computationally effective optimization procedure, and have demonstrated that it finds good minima of the objective function on both synthetic and real images.

3 Issues in Multi-Image Surface Reconstruction

In this section, we briefly discuss some of the key issues in multi-image surface reconstructions, and outline how we address the issues in this paper. These outlines will be expanded upon in Section 4.

3.1 Surface Shape and its Representation

Since the task is to reconstruct a surface from multiple images whose vantage points may be very different, we need a surface representation that can be used to generate images of the surface from arbitrary viewpoints, taking into account self-occlusion, self-shadowing, and other viewpoint-dependent effects. Clearly, a single image-centered representation, such as a depth map, is inadequate for this purpose. Instead, an object-centered surface representation is required.

There are many object-centered surface representations that are possible. However, there are some practical issues that are important in choosing an appropriate one. First, the representation should be general-purpose in the sense that it should be possible to represent any continuous surface, closed or open, and of arbitrary genus. Second, it should be relatively straightforward to generate an instance of a surface from standard data sets such as depth maps or clouds of points. Finally, there should be a computationally simple correspondence between the parameters specifying the surface and the actual 3-D shape of the surface, so that images of the surface can be easily generated, thereby allowing the integration of information from multiple images.

A hexagonally connected mesh of 3-D vertices, as in Figure 2, is an example of a surface representation that meets the criteria stated above, and is the one we have chosen for this paper. Such a mesh defines a surface composed of three-sided planar polygons that we call triangular facets, or simply facets. Triangular facets are particularly easy to manipulate for image and shadow generation, since they are the basis for many 3-D graphics systems. Hexagonal meshes can be used to construct virtually arbitrary surfaces. Finally, standard triangulation algorithms can be used to generate such a surface representation from real noisy data [18, 42].

3.2 Material Properties and their Representation

Objects in the world are composed of many types of material, and the material type can vary across the object's surface in many ways. The key issues, therefore, are the type of material we wish to consider, and how its variation across the surface is to be represented. In general, one can represent a material type by its reflectance function, which maps the wavelength distribution and orientation of a light source, the normal to the surface, and the viewing direction into an image color. This function is generally quite complex. However, there are reflectance functions that are not only much simpler, but are also quite common. Such functions are modeled using only one, or, at most, a few,

parameters. Consequently, one can accurately model the material properties of a surface by representing these parameters at every point on the surface.

Probably the simplest, and most common, such function is the Lambertian reflectance function. For grey-level images, this function not only has a single parameter, albedo, which is the ratio of incoming to outgoing light intensity, but the image intensity is independent of viewpoint. For this reason, we have chosen to restrict ourselves to Lambertian surfaces in this paper. However, because we use a full 3-D representation, a generalization to specular surfaces would be fairly straightforward.

Having chosen a specific reflectance function, the remaining issue is how to represent the spatially-varying parameter(s). In general, one needs to be able to represent independent parameter values at every point of the surface. In terms of the mesh representation of the surface, this implies some type of spatial sampling of each facet. Given the finite resolution of the images, and other practical considerations, we have chosen to use two types of spatial sampling. The first is most appropriate when the parameters vary quickly across the surface, and the second when they vary more slowly. For the former case, we use a uniform sampling of each facet, where the inter-sample spacing corresponds roughly to no more than one or two pixels in any of the images. For the later case, we use a single value associated with each facet.

As we shall see later, the two different representations are used somewhat differently, and the choice of which representation to use is made on a facet-by-facet basis as a function of the images.

3.3 Information Sources for Reconstruction

There are a number of information sources that are available for the reconstruction of a surface and its material properties. Here, we consider two classes of information.

The first class are those information sources that require a single image, such as texture gradients, shading, and occlusion edges. When using multiple images and a full 3-D surface rep-

resentation, however, we can do certain things that cannot be done with a single image. First, the information source can be checked for consistency across all images, taking into account occlusions. Second, the information can be "averaged" over all the images, when the source is consistent and occlusions are taken into account, to increase its sensitivity.

The second class are those information sources that require at least two images, such as the triangulation of corresponding points between input images (given camera models and their relative positions). Generally speaking, this source is most useful when corresponding points can be easily identified, and their image positions accurately measured. The ease and accuracy of this correspondence can vary significantly from place to place in the image set, and depends critically on the type of feature used. Consequently, whatever the type of feature used, one must be able to identify where in the images that feature provides reliable correspondences, and what accuracy one can expect.

The image feature that we have chosen for correspondence (though it is by no means the only one possible) is simply intensity, because the Lambertian reflectance model described earlier implies that the image intensity of a surface point is independent of the viewing direction. Therefore, corresponding points should have the same intensity in all images. Clearly, intensity can only be a reliable feature when the albedo varies quickly enough on the surface (and, consequently, the images are highly textured), and the search space is sufficiently narrow. Otherwise, there would be significant ambiguity in the correspondence of pixels across the images.

In contrast to our approach traditional correlation-based stereo methods use fixed-size windows in images, which can only yield correct results when the surface is tangential to the image plane. Instead, we compare the intensities as projected onto the facets of the surface, which is equivalent to having variable-shaped windows in the images. Consequently, if the original surface is well modeled by a mesh surface, the reconstruction can be significantly more accurate. The Hierarchical Warp Stereo System [39] is another example of a method that takes into account the variable shapes of windows required

for accurate reconstruction of a surface, though it uses only an image-centered representation of the surface.

As for the monocular information source, we have chosen to use shading. There are a number of reasons for this. First, we are using a Lambertian reflectance model, making shading a relatively simple source of information. Second, shading is most reliable when the albedo varies slowly across the surface, which is the natural complement to intensity correspondence, which requires quickly varying albedo. The complementary nature of these two sources should allow us to accurately recover the surface geometry and material properties for a wide variety of images.

In contrast to our approach traditional uses of shading information assume that the albedo is constant across the entire surface, which is a major limitation when applied to real images. We overcome this limitation by improving upon a method to deal with discontinuities in albedo alluded to in the summary of [30, 31]. We compute the albedo at each facet using the normal to the facet, a light-source direction, and the average of the intensities projected onto the facet from all images. Since we use the average of the projected intensities, this computed albedo minimizes the mean squared error between the images of the mesh surface and the input images. The variation of this computed albedo across the surface is the actual information source used to recover the surface. For example, if the albedo of the real surface were indeed constant, as in traditional shape-from-shading problems, then the measured variation in albedo will be zero for the correct mesh surface, and we will have recovered both surface shape and albedo. The distinct advantage of this approach over the traditional one is that it can deal with surfaces whose albedo is not constant, but instead varies slowly over the surface.

In the following subsection, we describe how these two sources of information are combined and used to reconstruct surfaces.

3.4 Combining and Using Information Sources

Simply put, our approach to surface reconstruction is to adjust the parameters of the surface (in the case of the mesh, this means the coordinates of the vertices), until the images of the surface are most consistent with the information sources described above. This approach requires a number of things. First, one must have an initial estimate of the surface. In this paper, this is derived from a standard correlation-based stereo algorithm. Second, one must know the light source direction, camera models, and their relative positions so that images of the surface can be generated (we assume these are provided *a priori*). Third, one must have a way of quantifying what is meant by "most consistent with the information sources." Here, we use an objective function that is a linear combination of components, one for each information source, whose weights are determined on a facet-by-facet basis as a function of the images. Finally, one must have a computationally effective means of finding a surface, given the initial estimate, that is reasonably close to the best of all possible surfaces according to the objective function.

Our combined objective function has three components, two of which were mentioned above: an intensity correlation component, and an albedo variation component. A third component is a measure of the smoothness of the surface. The first two components are weighted differently at each facet as a function of the image intensities projected onto the facet, while the surface smoothness component has the same weight everywhere, but is typically decreased as the iterations proceed.

Since the intensity correlation component depends on the difference in intensity at a given point, it is most accurate when the images are highly textured in the areas that the facet projects to. To see this, consider the case when the images have constant intensity in the neighborhood of the projected facet: the difference in intensity will be a constant, independent of small variations in the facet's position or orientation. On the other hand, when the images are highly textured, small changes in the facet

can significantly change the value of this component. Thus, we weight the intensity correlation component most strongly for those facets in which the projected image intensities are highly textured.

Conversely, the albedo variation component is most accurate when the intensities within a facet vary slowly. This is because we are assuming that the albedo varies slowly enough across the surface that a constant-albedo facet is a good model for the surface. Since the facets are planar, this should produce images whose intensities are constant within the projected facet. Thus, we weight the albedo variation component most strongly when the projected intensities within a facet vary slowly.

Since rapidly changing albedoes produce highly textured image regions, our weighting scheme, in effect, turns off the shading component and turns on the stereo component in such regions. Thus, it provides the shape from shading component with implicit boundary conditions at the edge of regions of constant albedo.

The surface smoothness component is required as a stabilizing term because neither of the above components is likely to be exactly correct, the surfaces are not exactly Lambertian, the camera positions are not exactly correct, there is noise in the images, and so on. Currently, we use the heuristic technique of starting with a relatively large weight for the smoothness component, and decrease it as the iterations proceed. The theoretically optimal point at which the smoothness weight should no longer be decreased is still an open question, although a single, empirically determined, value has been used with great success across all of the images presented in this paper when using all of the components.

In the following section, we describe the surface representation and optimization algorithm in more detail.

4 Details of Surface Model and Optimization Procedure

As discussed in the previous section, our approach to recovering surface shape and re-

flectance properties from multiple images is to deform a three-dimensional representation of the surface so as to minimize an objective function. The free variables of this objective function are the coordinates of the vertices of the mesh representing the surface, and the process is started with an initial estimate of the surface. For the experiments described in this paper, we have derived this initial estimate by triangulating the smooth depth-map generated by the correlation-based stereo algorithm described in [19, 15]. Figure 1 illustrates the output of this algorithm on a synthetic stereo pair.

Alternatively, we could have relied on more sophisticated algorithms that can triangulate noisy laser or stereo range-data to derive our initial estimates [14, 18, 42]. All these methods tend to smooth the data and to interpolate blindly in the absence of data so that their output needs to be refined by algorithms such as ours.

In this section, we describe more formally each part of our approach.

4.1 Images and Camera Models

In this paper, we assume that images are monochrome, and that their camera models are known *a priori*. The set of grey-level images is denoted $G = (g_1, g_2, \dots, g_{n_g})$. A point in an image is denoted $u = (u, v)$, and the intensity of point u in image g_i is denoted $g_i(u)$. For non-integer values of u we use bilinear interpolation over the four points represented by the floor and ceiling of the coordinates of u .

The projection of an arbitrary point $x = (x, y, z)$ in space into image g_i is denoted $m_i(x)$. There are well-known methods for correcting both geometric and radiometric errors in images, as surveyed in [4], pp. 68–77. Thus, we assume that all effects of lens distortion and the like have been taken care of in producing the input images, so that the projection of a surface into an image is well modeled by a perspective projection. Thus, $u = m_i(x)$ can be written as:

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = M_i \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

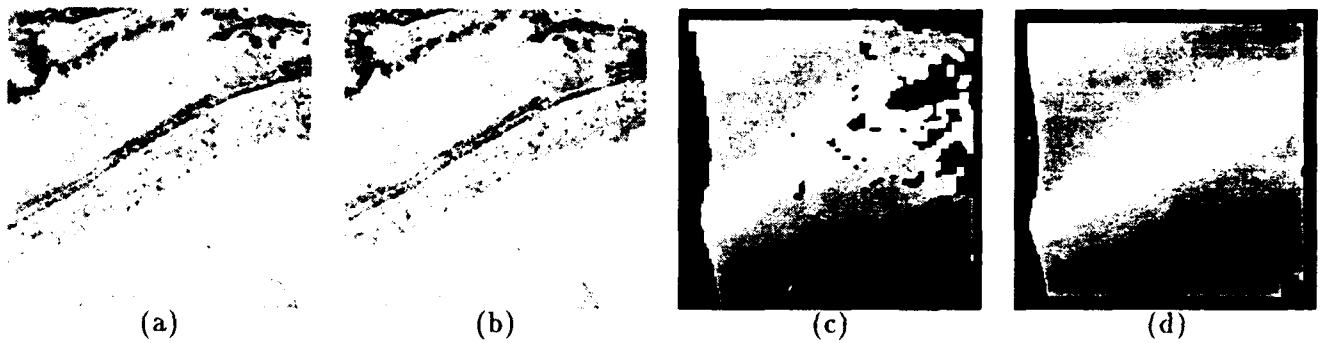


Figure 1: (a,b) A synthetic stereo pair generated by texture-mapping a real image of the Martin-Marietta ALV test-site onto a Digital Elevation Model (DEM). (c) The disparity map using a correlation-based algorithm. The black areas indicate that the stereo algorithm could not find a match. Elsewhere, lighter greys indicate higher elevations. (d) The same disparity map after smoothing and interpolation.

$$\begin{aligned} u &= U/W \\ v &= V/W, \end{aligned}$$

where M_i is a three by four projection matrix.

4.2 Surface Representation

We represent a surface \mathcal{S} by a hexagonally-connected set of vertices $\mathbf{V} = (v_1, v_2, \dots, v_{n_v})$ called a *mesh*. The position of vertex v_j is specified by its Cartesian coordinates (x_j, y_j, z_j) . Figure 2 shows such a mesh as a wire frame and as a shaded solid surface.

Each vertex in the interior of the surface has exactly six neighbors. The neighbors of vertex v_j are consistently ordered in a clock-wise fashion. Vertices on the edge of a surface may have anywhere from two to five neighbors.

Neighboring vertices are further organized into triangular planar surface elements called *facets*, denoted $\mathbf{F} = (f_1, f_2, \dots, f_{n_f})$. The vertices of a facet are also ordered in a clock-wise fashion. In this work, we require that the initial estimate of the surface have facets whose sides are of equal length. The objective function described below tends to maintain this equality, but does not strictly enforce it. The representation can be extended in a straight-forward fashion to support different surface resolutions by sub-dividing facets (which we have done). However, facets of a given resolution will still be required to have approximately equal sides.

4.3 Objective Function

The objective function $\mathcal{E}(\mathcal{S})$ that we use to recover the surface is best described in two equations. In the first equation,

$$\mathcal{E}(\mathcal{S}) = \lambda_D \mathcal{E}_D(\mathcal{S}) + \mathcal{E}_G(\mathcal{S}), \quad (1)$$

$\mathcal{E}(\mathcal{S})$ is decomposed into a linear combination of two components. The first component, $\mathcal{E}_D(\mathcal{S})$, is a measure of the deformation of the surface from a nominal shape, and is independent of the images. For this paper, the nominal shape is a plane. Higher-order measures, such as deformation from a sphere, are also possible. This nominal shape represents the shape that the surface would take in the absence of any information from the images.

The second component,

$$\mathcal{E}_G(\mathcal{S}) = \lambda_C \mathcal{E}_C(\mathcal{S}) + \lambda_S \mathcal{E}_S(\mathcal{S}) \quad (2)$$

depends on the images, and is the one that drives the reconstruction process. It is further decomposed into a linear combination of the two information sources described in the previous section: a multi-image correlation component, $\mathcal{E}_C(\mathcal{S})$, and a component that depends on the shading of the surface, $\mathcal{E}_S(\mathcal{S})$.

These components, and their relative weights, are described in more detail below.

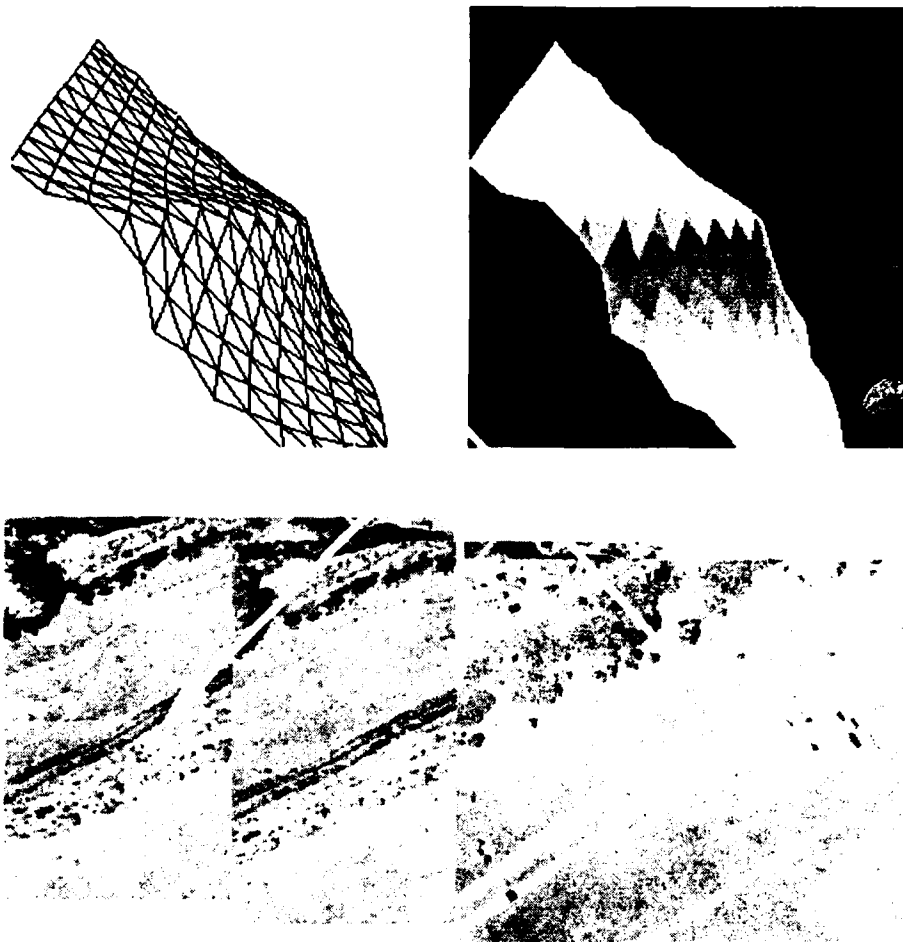


Figure 2: The top row shows a hexagonal mesh as both a wireframe and a shaded surface. The bottom row shows several images of a scene. In our approach, these images are projected onto the mesh using camera models.

4.3.1 Surface Deformation Component

As stated earlier, the surface deformation (or smoothness) component is a measure of the deviation of the mesh surface from some nominal smooth shape. When the nominal shape is a plane, we can approximate this as follows.

Consider a perfectly planar hexagonal mesh for which the distances between neighboring vertices are exactly equal. Recall that the mesh is defined so that the neighbors of a vertex v_i are ordered in a clock-wise fashion, and are denoted $v_{N_i(j)}$. If the hexagonal mesh was perfectly planar, then the third neighbor over from the j^{th} neighbor, $v_{N_i(j+3)}$, would lie on a straight line with v_i and $v_{N_i(j)}$. Given that the inter-vertex distances are equal, this implies that coordinates of v_i equal the average of the coordinates

of $v_{N_i(j)}$ and $v_{N_i(j+3)}$, for any j .

Given the above, we can write a measure of the deviation of the mesh from a plane as follows:

$$\mathcal{E}_D(\mathcal{S}) = \sum_{i=1}^{n_v} \sum_{\substack{j=1 \\ k=N_i(j) \\ k'=N_i(j+3)}}^3 \frac{(2x_i - x_k - x_{k'})^2 + (2y_i - y_k - y_{k'})^2 + (2z_i - z_k - z_{k'})^2}{\dots}$$

Note that this term is also equivalent to the squared directional curvature of the surface when the sides have approximately equal lengths [27]. Also, this term can accommodate multiple resolutions of facets by normalizing each term by the nominal inter-vertex spacing.

ing of the facets.

4.3.2 Multi-Image Intensity Correlation

The multi-image intensity correlation component is the sum of squared differences in intensity from all the images at a given sample-point on a facet, summed over all sample-points, and summed over all facets. This component is presented in stages in the remainder of this subsection.

First, we define the sample-points of a facet by taking advantage of the fact that all points on a triangular facet are a convex combination of its vertices. Thus, we can write the sample-points $\mathbf{x}_{k,l}$ of facet f_k as:

$$\mathbf{x}_{k,l} = \lambda_{l,1} \mathbf{x}_{k,1} + \lambda_{l,2} \mathbf{x}_{k,2} + \lambda_{l,3} \mathbf{x}_{k,3}, \quad l = 3, 4, \dots, n_s,$$

where $\mathbf{x}_{k,1}$, $\mathbf{x}_{k,2}$, and $\mathbf{x}_{k,3}$ are the coordinates of the vertices of facet f_k , and $\lambda_{l,1} + \lambda_{l,2} + \lambda_{l,3} = 1$. In the top half of Figure 3(a), we see an example of the sample points of a facet.

Next, we develop the sum of squared differences in intensity from all images for a given point \mathbf{x} . Recall that a point \mathbf{x} in space is projected into a point \mathbf{u} in image g_i via the perspective transformation $\mathbf{u} = \mathbf{m}_i(\mathbf{x})$. Consequently, the sum of squared differences in intensity from all the images, $\sigma'(\mathbf{x})$, is:

$$\begin{aligned} \mu'(\mathbf{x}) &= \frac{1}{n_i} \sum_{i=1}^{n_i} g_i(\mathbf{m}_i(\mathbf{x})) \\ \sigma'(\mathbf{x}) &= \frac{1}{n_i} \sum_{i=1}^{n_i} (g_i(\mathbf{m}_i(\mathbf{x})) - \mu'(\mathbf{x}))^2 \end{aligned}$$

Figure 3(a) illustrates the projection of a sample-point of a facet onto several images.

The above definition of $\sigma'(\mathbf{x})$ does not take into account occlusions of the surface. To do so, we use a "Facet-ID" image shown in Figure 4. It is generated by encoding the index i of each facet f_i as a unique color, and projecting the surface into the image plane using a standard hidden-surface algorithm. Thus, when a sample-point from facet f_k is projected into an image, the index k is compared to the index stored in the Facet-ID image at that point. If they are the same, then the sample-point is

visible in that image, otherwise, it is not. Let $v_i(\mathbf{x}) = 1$ when point \mathbf{x} is determined to be visible in image g_i by the method above, and $v_i(\mathbf{x}) = 0$ otherwise. Then, the correct form for the sum of squared differences in intensity at a point \mathbf{x} is:

$$\begin{aligned} \mu(\mathbf{x}) &= \frac{\sum_{i=1}^{n_i} v_i(\mathbf{x}) g_i(\mathbf{m}_i(\mathbf{x}))}{\sum_{i=1}^{n_i} v_i(\mathbf{x})} \\ \sigma(\mathbf{x}) &= \frac{\sum_{i=1}^{n_i} v_i(\mathbf{x}) (g_i(\mathbf{m}_i(\mathbf{x})) - \mu(\mathbf{x}))^2}{\sum_{i=1}^{n_i} v_i(\mathbf{x})} \end{aligned}$$

Finally, summing $\sigma(\mathbf{x})$ over all sample-points and over all facets yields the multi-image intensity correlation component:

$$\mathcal{E}_C(\mathcal{S}) = \sum_{k=1}^{n_f} c_k \sum_{l=3}^{n_s} \sigma(\mathbf{x}_{k,l}),$$

where c_k is a number between 0 and 1 that weights the contribution from each facet differently, depending on the average degree of texturing within a facet (see Section 4.3.4).

When the original surface giving rise to the images is sufficiently textured, this component should be smallest when the surface \mathcal{S} closely approximates the original surface. However, when the surface has constant, or nearly constant, albedo this component would be small for many different surfaces. As an extreme example of this ambiguity, consider a planar surface with constant albedo. This produces images with constant intensity. Thus, this component will not be able to constrain the shape of the surface, since the difference in intensity will be zero for all surfaces.

An example of using only the intensity-correlation and smoothness components on the synthetic stereo pair of Figure 1 is shown in Figure 5. The top row of the figure depicts the initial surface estimate. Figures 5(a) and (b) are shaded images of the mesh. Figure 5(c) depicts the error from ground-truth elevation for the left image, where black indicates zero error, and white indicates an error corresponding to a few pixels in disparity. Figure 5(d) depicts the squared difference in intensity between the left image and the right images warped using the disparity map. Note that the worst errors occur

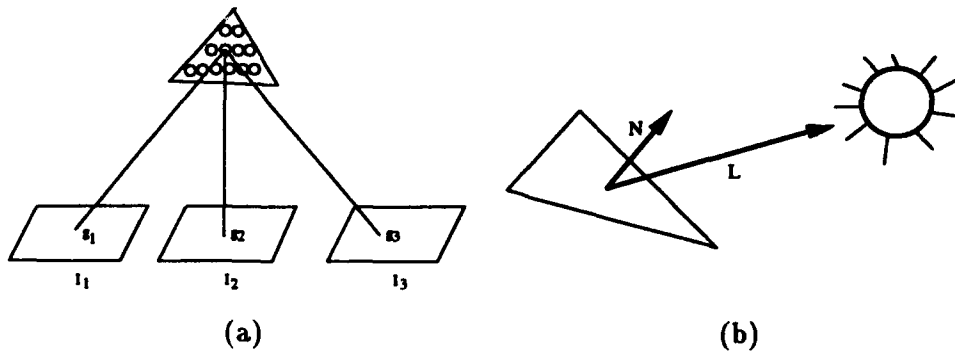


Figure 3: (a) Facets are sampled at regular intervals as illustrated here. We use the grey levels of the projections of these sample points to compute the stereo score. (b) The albedo of each facet is estimated using the facet normal \vec{N} , the light source direction \vec{L} and the average grey level of the projection of the facet into the images.

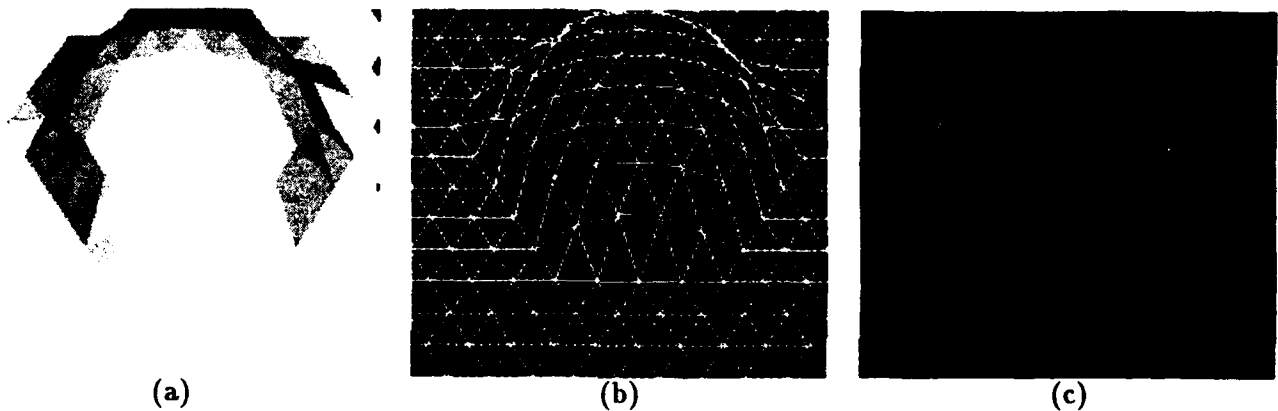


Figure 4: Illustration of the projection of a mesh, and the "Facet-ID" image used to accommodate occlusions during surface reconstruction. (a) A shaded image of a mesh. (b) A wire-frame representation of the mesh (bold white lines) and the sample-points in each facet (interior white points). (c) The "Facet-ID" image, wherein the color at a pixel is chosen to uniquely identify the visible facet at that point (shown here as a grey-level image).

along the steep ridge of the terrain, where the constant-depth assumption of correlation-based stereo is most strongly violated.

The bottom row of Figure 5 illustrates the result of the optimization procedure, described in Section 4.4, using only the intensity-correlation and smoothness components. Note that the overall error in both elevation and intensity is lower, and that the error is no longer concentrated along the ridge. As a result, the ridge is clearly sharper in the shaded views.

4.3.3 Shading

The shading component of the objective function is the sum, over all facets, of the difference between the computed albedo of the facet and the computed albedoes of all of its neighbors. The motivation for this component, and its precise form, follow.

Recall that the Lambertian reflectance model defines the intensity g at a point on a surface with a unit surface normal \vec{N} as:

$$g = \alpha(a + b\vec{N} \cdot \vec{L}), \quad (3)$$

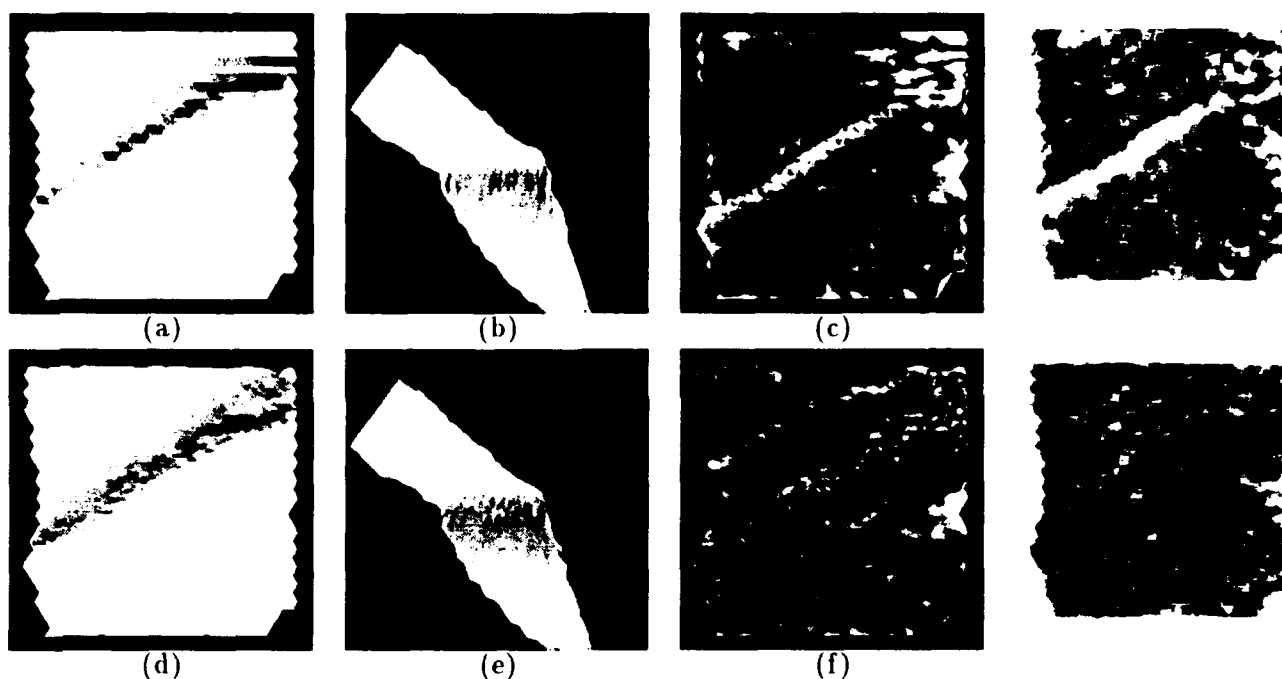


Figure 5: (a,b) Two shaded views of the mesh derived from the smoothed disparity map of Figure 1(d). (c) Deviations in altitude from the elevation data used to generate the synthetic pair. (d) Intensity error image, created by warping the right image into the left image using the disparities corresponding to the elevations of the mesh facets and computing the squared difference between these two images (e,f,g,h) Corresponding images after stereo optimization. Note that the ridge now appears much sharper in the shaded views, and that the overall error is smaller and more evenly distributed.

where α is the albedo of the surface, a is the magnitude of the ambient light, b is the magnitude of a point light source, and \vec{L} is the direction of the point light source as depicted in Figure 3(b).

Note that g is independent of the viewing direction. Consequently, if we were to image a planar Lambertian facet from several points of view, its intensity would be the same for all pixels in the projection of the facet. Conversely, if we were to measure the average intensity \bar{g}_k of all of the pixels within the projection of a facet f_k , we could compute its albedo, α_k , as follows:

$$\alpha_k = \frac{\bar{g}_k}{(a + b \vec{N} \cdot \vec{L})}. \quad (4)$$

This assumes, of course, that the facet is well-modeled by a single albedo, and that the variation in intensity is due only to noise. In this paper, we assume that the ambient and direct illumination (*i.e.*, a , b , and \vec{L}) are given, al-

though some of these parameters could be included in the optimization, as was done in [31].

The average intensity \bar{g}_k of a facet is computed by scanning over all the Facet-ID images for index k , and taking the average of the intensities at matching points in the corresponding images. This method provides an inexpensive way of computing the average intensity while taking occlusions into account.

Now, if the original surface had exactly constant albedo, and if our mesh surface were a good approximation to the original surface, then the computed albedoes should be approximately the same across all facets. Thus, some measure of the variation in computed albedoes would be a good measure of the correctness of the mesh surface. If the albedo varies slowly across the surface, we propose that an appropriate measure of this variation is the difference between the computed albedo at the facet and the computed albedoes of all of its neighbors:

$$\mathcal{E}_S(\mathcal{S}) = \sum_{k=1}^{n_f} (1 - c_k) \sum_{j \in N_f(k)} (1 - c_j) (\alpha_k - \alpha_j)^2,$$

where $N_f(k)$ is the set of indices of the facets that are neighbors of facet f_k , and c_k and c_j are numbers between 0 and 1 that depend on the degree of texturing within facets f_k and f_j .

An example of using only the shading and smoothness components is illustrated in Figure 6. Figure 6(a) shows a shaded view of the original surface, a hemisphere with constant albedo. Figures 6(b) and (c) show shaded views of the initial surface estimate, which was derived by adding white noise to the vertex coordinates of the original surface. Figures 6(d) and (e) are the shaded views of the result after optimization, and Figure 6(f) is the albedo map for the surface, i.e. the intensity in the image represents the albedo of the surface. Note that the albedo and shape are well recovered except near the edge of the hemisphere where the image intensity varies rapidly across the image. This is because the approximation we use in the derivatives of this component is that the mean intensity within a facet does not vary significantly in the neighborhood of a facet, which is violated for facets that straddle the boundary. This does not hurt us when combining shading with the stereo component since, as explain in the following subsection, we turn off the shading component in such areas.

4.3.4 Combining the Components

Recall that the objective function $\mathcal{E}(\mathcal{S})$ is a linear combination of three components:

$$\mathcal{E}(\mathcal{S}) = \lambda_D \mathcal{E}_D(\mathcal{S}) + \lambda_C \mathcal{E}_C(\mathcal{S}) + \lambda_S \mathcal{E}_S(\mathcal{S}),$$

where the last two components are themselves linear combinations of subcomponents computed on a per-facet basis:

$$\begin{aligned} \mathcal{E}_C(\mathcal{S}) &= \sum_{k=1}^{n_f} c_k \sum_{l=3}^{n_s} \sigma(\mathbf{x}_{k,l}) \\ \mathcal{E}_S(\mathcal{S}) &= \sum_{k=1}^{n_f} (1 - c_k) \sum_{j \in N_f(k)} (1 - c_j) (\alpha_k - \alpha_j)^2. \end{aligned} \quad (5)$$

Thus, one needs to specify both the λ s, defining the relative weights of the components, and the c_k s, defining the relative weights of the facets in each of these components.

The λ weights are defined as follows:

$$\begin{aligned} \lambda_D &= \frac{\lambda'_D}{\|\vec{\nabla} \mathcal{E}_D(\mathcal{S}^0)\|} \\ \lambda_C &= \frac{\lambda'_C}{\|\vec{\nabla} \mathcal{E}_C(\mathcal{S}^0)\|} \\ \lambda_S &= \frac{\lambda'_S}{\|\vec{\nabla} \mathcal{E}_S(\mathcal{S}^0)\|}, \end{aligned} \quad (6)$$

where \mathcal{S}^0 is the initial estimate of the surface, and the λ 's are user defined weights. Normalizing each component by the magnitude of its initial gradient allows the components to have roughly the same influence when the λ 's are equal. Thus, the user can more easily specify the relative contributions of each component in an image-independent fashion. This normalization scheme was used with great success in [16], and is analogous to standard constrained optimization techniques in which the various constraints are scaled so that their eigenvalues have comparable magnitudes [34].

As mentioned earlier, the c_k weights are a function of the degree of texturing in the intensities projected within a facet f_k . A simple measure of the degree of texturing within a facet is the variance in intensity of all the pixels projecting onto the facet, denoted $\sigma_k(\mathcal{S})$ (using the Facet-ID image to accommodate occlusions). We have found that using the logarithm of $\sigma_k(\mathcal{S})$ yields the most stable results:

$$c_k = a \log(1 + \sigma_k(\mathcal{S})) + b, \quad (7)$$

where a and b are normalizing factors chosen so that the smallest c_k is zero, and the largest is one.

4.4 The Optimization Procedure

The purpose of the optimization procedure is to iteratively modify the surface \mathcal{S} so as to minimize $\mathcal{E}(\mathcal{S})$, given some initial estimate \mathcal{S}^0 , and some value for the weights λ'_S , λ'_C , and λ'_D

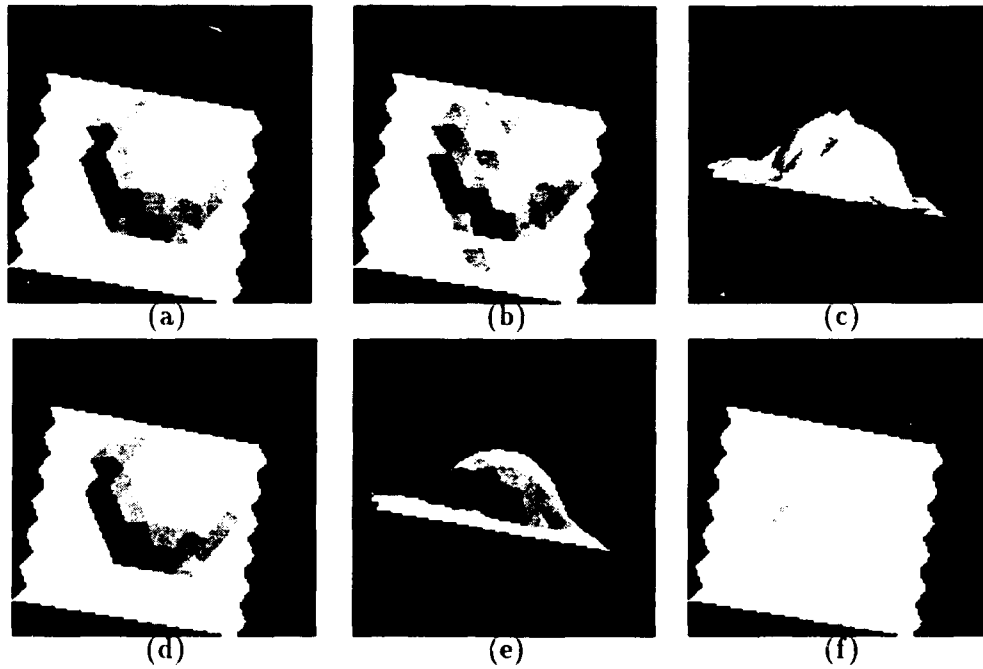


Figure 6: (a) Shaded image of a hemisphere of constant albedo. (b,c) Shaded views of randomized hemisphere used as a starting point. (d,e) Shaded views of the same hemisphere after optimization using only the shading component of the objective function. (f) The recovered albedo map.

(where $\lambda'_S + \lambda'_C + \lambda'_D = 1$) defined in Equation 7. Ideally, one would like to use as small a value of the deformation weight λ'_D as possible so as to minimize the bias introduced by this term. However, in practice, λ'_D serves a dual purpose. First, since the surface deformation term is a quadratic function of the vertex coordinates, it “convexifies” the energy landscape and improves the convergence properties of the optimization procedure. Second, as will be discussed in the results section, in the absence of a smoothing term, the objective function may overfit the data and wrinkle the surface excessively. Furthermore, the c_k weights of Equations 6 and 7 are computed for the initial position of the mesh and are only meaningful when it is relatively close to the actual surface.

Consequently, we use an optimization method that is inspired by the heuristic technique known as a continuation method [43, 28, 29, 30]. We first “turn off” the shading term by setting λ'_S (equation 7) to 0 and set λ'_D to a value that is large enough to sufficiently convexify the energy landscape but small enough to allow cur-

vature in the surface. In this paper we take the initial value of λ'_D to be 0.5. Given the initial estimate S^0 , a local minimum of this approximate objective function is found using a standard optimization procedure. Then, λ'_D is decreased slightly, and the optimization procedure is applied again, starting at the local minimum found for the previous approximation. This cycle is repeated until λ'_D is decreased to the desired value. Finally we “turn on” the shading term, compute the c_k weights and reoptimize. In all examples shown in the result section we use $\lambda'_C = \lambda'_S = .4$ and $\lambda'_D = .2$.

The stereo component effectively uses only first order information about the surface (i.e., the position of the vertices), whereas shading uses second order information about the surface (i.e., its surface normals). Thus, by optimizing the stereo component first, we effectively compute the zero order properties of the surface and set up boundary conditions that the shading component can then use to compute the first order properties of the surface in textureless regions. In section 5, we will show that this

leads to a significant improvement over using the stereo component alone.

When dealing with surfaces for which motion in one direction leads to more dramatic changes than motions in others, as is typically the case with the z direction in Digital Elevation Models (DEMs), we have found that the following heuristic to be useful. We first fix the x and y coordinates of vertices and adjust z alone. Once the surface has been optimized, we then allow all of the coordinates to vary simultaneously.

The optimization procedure we use at every stage is a standard conjugate-gradient descent procedure called FRPRMN (from [38]) in conjunction with the a simple line search algorithm. The conjugate-gradient procedure requires three inputs: 1) a function that returns the value of the objective function for any S ; 2) a function that returns the gradient of $\mathcal{E}(S)$, i.e., a vector whose elements are the partial derivatives of $\mathcal{E}(S)$ with respect to the vertex coordinates, evaluated at S ; and 3) an initial estimate S^0 .

The gradient of $\mathcal{E}(S)$ is conceptually straightforward, but is fairly complicated to derive manually. We have used the Maple¹ mathematical package to derive some of the terms. We summarize the calculation of the derivatives below in general terms.

The derivatives of the stereo term are linear combinations of image intensity derivatives and of derivatives of the 3-D projections of points onto the images. Since we use bilinear-interpolation of image values, the first derivatives of image intensity are linear combinations of the image intensities in the immediate neighborhood of the projection. Since sample-points are linear combinations in projective space of the mesh vertices, their projections are ratios of linear combinations of the projections of the vertices, which themselves depend linearly on the vertex coordinates. Consequently, the derivatives of these projections are ratios of linear combinations of the vertex coordinates and squares of linear combinations of the vertex coordinates.

Similarly, the derivatives of the shading term depend of the derivatives of the surface nor-

mal, which can be easily derived analytically, and from the derivative of the mean grey-level in the facets. In this work, the shading term is used mainly in the fairly uniform areas where the latter derivative is assumed to be small and therefore neglected.

5 Behavior of the Objective Function and Results

In previous sections, we have shown results of the optimization procedure using only one or the other of the image components of the objective function. In this section, we first illustrate the behavior of the complete objective function using synthetic data. We then show that the same behavior can be observed with real data, allowing us to generate accurate 3-D reconstructions of real surfaces from multiple images.

5.1 Synthetic Data

To demonstrate the properties of the objective function of Equation 1 and the influence of the coefficients defined in Equation 4, we use as input the five synthetic images of a shaded hemisphere with variable albedo shown at the bottom of Figure 7, both with and without the addition of white noise. Each column of the figure illustrates the steps used in the creation of the image at the bottom of the column. We begin with a mesh and an albedo map, shown in the top row. Then, for each view, two images are produced. The first image (second row of the figure) is the albedo map texture-mapped onto the mesh from the final image's point of view. The second image (third row of the figure) is a shaded view of the mesh, using a constant albedo equal to one. The final image is the point-by-point product of these two images because, by Equation 3, the imaged intensity of a Lambertian surface is the product of the albedo (first image) and the inner product of the light source and the surface normal (second image).

Figure 8 depicts graphically the result of our experiments. In each experiment we randomized the mesh by adding random numbers to the coordinates of the mesh vertices, and added different amounts of noise to the input images.

¹Trademark, Waterloo Maple Software

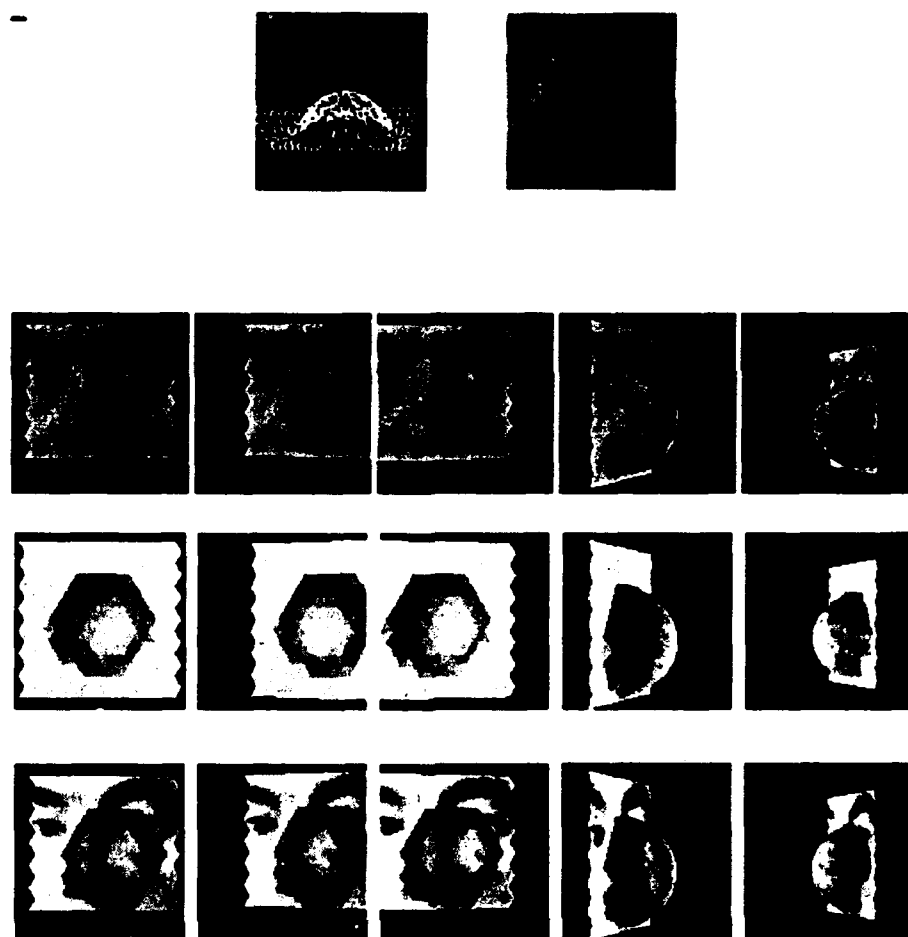


Figure 7: The making of synthetic images of a shaded hemisphere with variable albedo that conforms to our Lambertian model.

We then used our optimization procedure to estimate the true hemispherical shape and true albedo map. More precisely, starting from our randomized initial estimate, we first use stereo alone and progressively decrease the value of the λ'_D parameter of Equation 7 from 0.5 to 0. We then turn on the shading term by setting both λ'_D and λ'_S to 0.4, compute the c_k s of Equation 7 and optimize the full objective function. To show the stability of the process, we then recompute the c_k s for the optimized mesh and perform a second optimization using the updated values.

The first column of Figure 8 is for experiments using only the first, second, and third images from Figure 7, where there is little self occlusion. The second column is for experiments using the first, fourth, and fifth images, where

there is a significant amount of self occlusion. Finally, the third column is for experiments using all five images. In this particular set of experiments, we fixed the boundaries of the mesh and allowed only the z coordinates of the vertices to vary. However, the same overall behaviors can be observed without the boundary conditions.

The first row from the top of Figure 8 is a graph of the average squared error in elevation (the abscissa) versus decreasing λ'_D (the ordinate). To the left of the dotted vertical line, only the intensity correlation component is used. To the right, both the intensity correlation and shading components are used. The different curves are for different amounts of noise in the input images. The bottom curve is when there is no noise (other than quantization error),

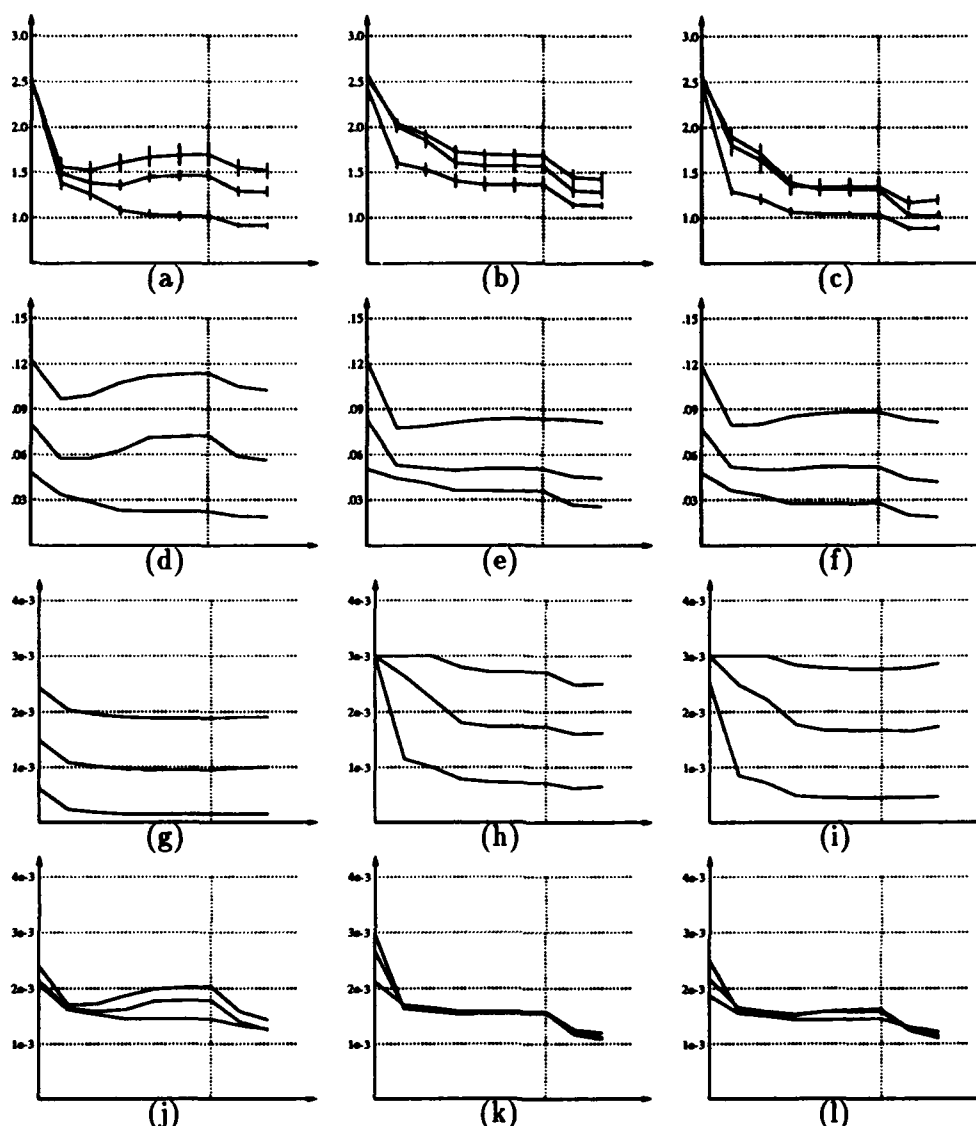


Figure 8: Graphs of the errors and objective function components while fitting a surface model to the synthetic shaded hemisphere images of Figure 7 (These graphs are explained in detail in the text.). (a,b,c) Average error in recovered elevation. (d,e,f) Average error in recovered albedo. (g,h,i) Stereo component of the energy. (j,k,l) Shading component of the energy.

the middle curve is for a noise variance of 4% of the image dynamic range, and the top curve is for a noise variance of 8%. The short vertical lines along the curves indicated the standard deviation of the average error over the 20 experiments performed to derive each curve.

The second row of Figure 8 is a graph of the average error in computed albedo. The third row is the average value of the intensity correlation component, $\mathcal{E}_C(\mathcal{S})$, and the fourth row is the average value of the shading component,

$\mathcal{E}_S(\mathcal{S})$.

Note that, as λ'_D decreases and stereo alone is used (i.e., as the ordinate is traversed rightwards to the dotted vertical line), the average elevation error decreases when there is no noise in the input image (bottom curve), as does the average albedo error and the two components of the objective function. However, when the images are noisy, the elevation error (first row) stops decreasing and may even begin to increase as we start fitting to the grey-level

noise, even though the value of the intensity correlation component (third row) continues to decrease (as it must). Furthermore, both the albedo error (second row) and the shading component (fourth row) also begin to increase when the elevation error does. This is natural since for smaller values of λ'_D the surface becomes rougher and its normals less well-behaved. As a result, the estimated albedoes of Equation 4 become less reliable and noisier.

In other words, an increase in the shading component provides us with a warning that we are starting to overfit the data. This is a valuable behavior in itself. Furthermore, by turning on the shading component of our objective function (those parts of the graphs that are to the right of the vertical dotted line), we can bring down both the error in albedo and the value of albedo component with at worst of modest increase in the value of the stereo component, resulting in an overall reduction of the elevation error. Even when there is nothing but quantization noise in the image, the addition of the shading component can make a small, but still noticeable difference. The reasons for this are twofold:

1. The shading component averages over whole facets and is therefore less sensitive to uncorrelated noise.
2. The shading component uses absolute intensity values whereas the stereo component uses intensity differences. Thus, in the presence of noise in textureless areas, the signal-to-noise ratio for the absolute values (used by the shading component) is larger than for the differences (used by the stereo component), thereby making the shading term more robust.

However, in our experience, the shading term can only be used reliably when the surface is relatively close to the correct answer. This is not surprising since the stereo deals directly with elevations whereas shading deals with derivatives of elevation. Consequently we have chosen the optimization "schedule" described above where we first optimize using stereo alone and turn on shading only later.

There is another important point to note about these results. The elevation errors in the second row, i.e. those generated using images 1, 4, and 5 with a lot of self occlusion are very close to those of the first row, i.e. those generated using images 1, 2, and 3 with little self occlusion, while those in the final row (using all five images) are significantly better. Furthermore, in this particular case, the results for images 1,4 and 5 are even slightly better than those for images 1,2 and 3 in the presence of noise because the former correspond to larger baselines. In other words, having the same number of images, but with significant self-occlusions, does not hurt our procedure. However, adding new images that contain significant self-occlusions actually improves the results.

We now turn to real images and show that the same properties can also be observed there.

5.2 Real Images

In Figure 9 we show the result of running the stereo component of our objective function on a real stereo pair corresponding to the same site as the synthetic images of Figure 1. Note that the radiometry of the left and right images are actually slightly different. We correct for this by first band-passing each image by taking the difference between the image and its gaussian convolution. This is approximately equivalent to replacing the simple correlation that our objective function uses by a normalized correlation, but is computationally more efficient. We then applied the optimization using exactly the same schedule and parameters as in the synthetic case, with the exception that λ_S is not reduced quite as much for the real images as for the synthetic ones in the first step of the procedure. Note that the recovered ridge is even sharper than in the synthetic case. This is because the Digital Elevation Model used to produce the synthetic right image was actually a slightly smoothed version of the terrain, in which one side of the ridge is an almost vertical cliff. Thus, even though we do not currently have ground truth for the real case, the sharpness of the recovered cliff, which matches what is seen using a stereoscope, leads us to believe that the algorithm has performed well.

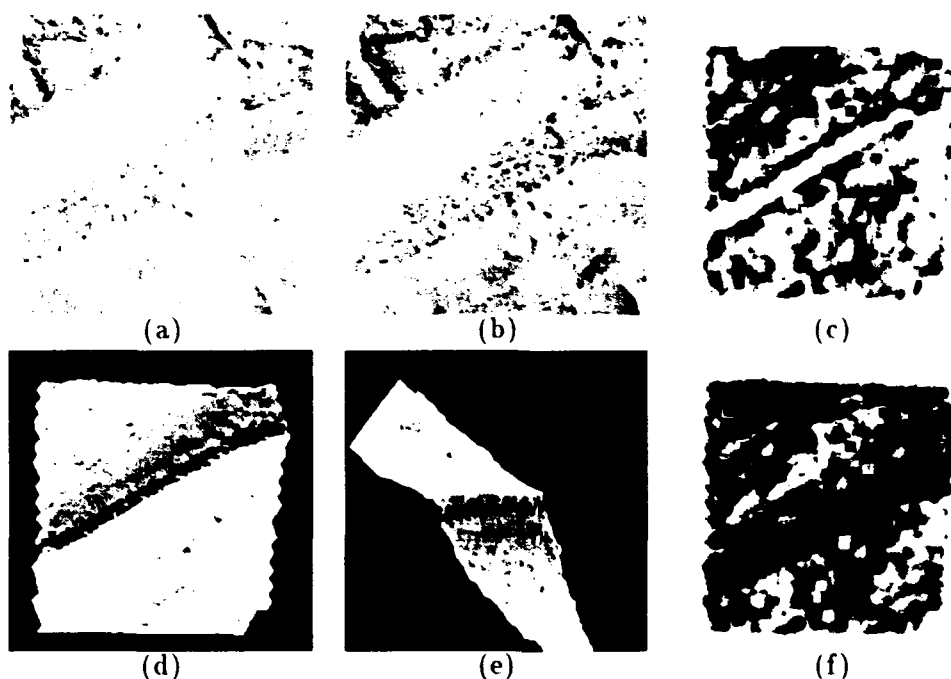


Figure 9: (a,b) A stereo pair of real images of the Martin-Marietta ALV test-site used in Figure 1. (c) Intensity error image computed using the method described in Figure 1(c) (d,e) Shaded views of the mesh after optimization. (f) Intensity error image after optimization. Note that the ridge is now very sharp. This corresponds accurately to the almost vertical cliff that can be seen when viewing the stereo pair with a stereoscope.

In Figure 10 we show three triplets of images of faces. They have been produced using the INRIA three camera system [13] that provides us with the 3 by 4 projection matrices we need to perform our computations. In this case it is essential to have more than two images to be able to reconstruct both sides of the face because of self-occlusions. For each triplet, we have computed disparity maps corresponding to images 1 and 2 and to images 1 and 3 and combined them to produce the depth maps shown in the rightmost column of the figure using the algorithms described in [19, 15].

The depth maps have then been smoothed and triangulated to produce the initial surfaces shown in the upper left corner of Figures 11, 12, and 13. In the first row of these three figures, we show the result of the optimization using stereo alone as we progressively decrease the smoothness constraint and allow all three vertex coordinates to be adjusted. Note that for the first two triplets (Figures 11 and 12), we recover more and more detail until the sur-

face eventually starts to wrinkle, without apparent improvement in accuracy. The third triplet poses an even more difficult problem: there are strong specularities on both the forehead and the nose that strongly violate our Lambertian model. Because there are very few other points that can be matched on the nose, the algorithm latches on to these specularities and yields a poor result.

In the bottom row of Figures 11, 12, and 13, we show our final results obtained by turning on the shading term and reoptimizing the meshes. For these images we did not know a priori the light source-direction, we therefore estimated it by choosing the direction that minimizes the shading component of the objective function given the surface optimized using only the stereo component. In all three images, the main features of the faces, nose, mouth and eyes have been correctly recovered. The improvement is particularly striking in the case of the face in Figure 13. The shading component was able to achieve this result because it uses

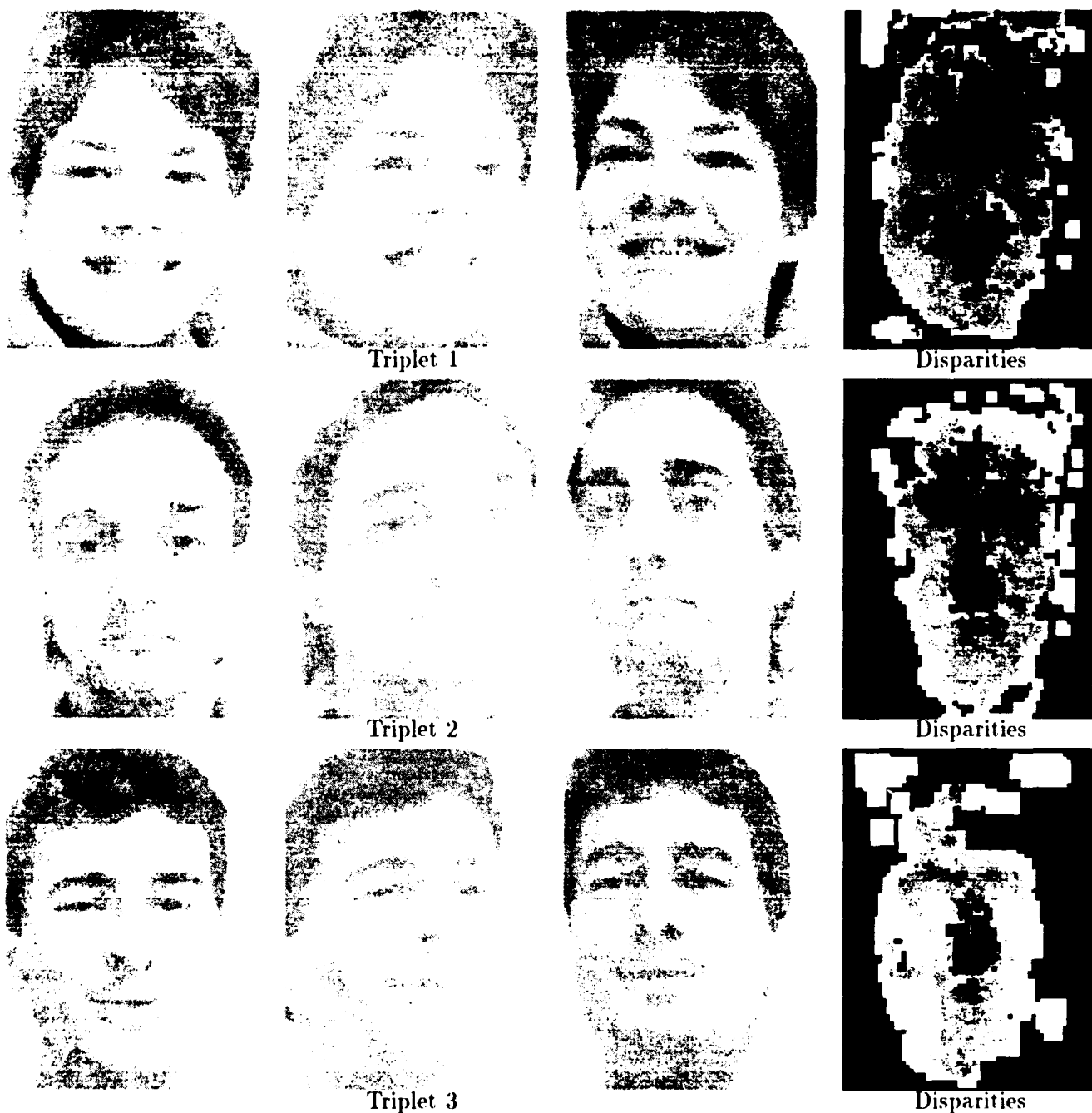


Figure 10: Triplets of face images and corresponding disparity maps (courtesy of INRIA).

the monocular information around the specularities. The stereo component cannot take advantage of the information around the specularities because very few points are visible in at least two images simultaneously, and because there is little texture. Of course, the effect of the specularities has not completely disappeared (there is indeed still a small artifact on the nose) but

has been outweighed by the surrounding information. A more principled approach to solving this problem would be to explicitly include a specularity term in our shading model.

The graphs of Figure 14 depict the behavior of the stereo and shading components of the objective function for the three triplets. The four values of the scores to the left of the thick

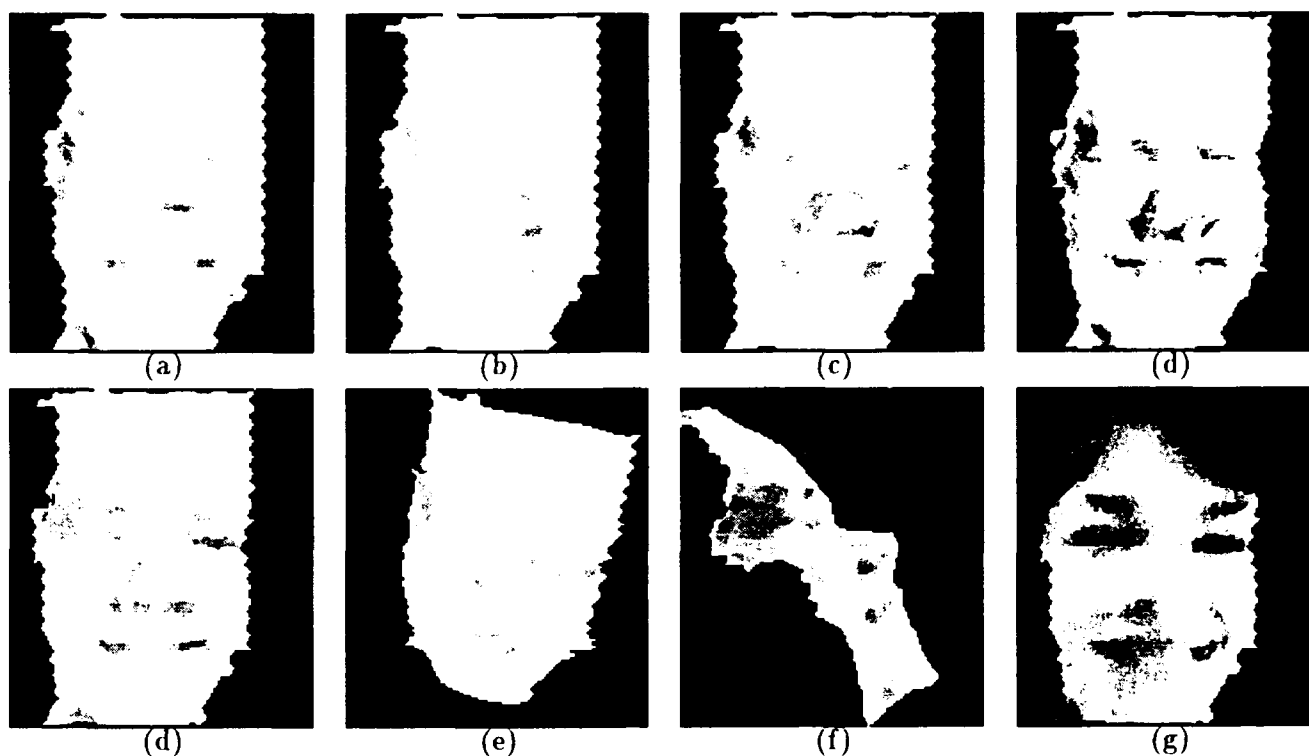


Figure 11: Results for the first triplet of Figure 10. (a) Shaded view of the mesh generated by smoothing and triangulating the computed disparity map. We use it as the starting condition for our optimization procedure. (b,c,d) The mesh after optimization using only the stereo term, with progressively less smoothing. (e,f,g) Several views of the mesh after optimization using both stereo and shading. (h) The recovered albedo map.

dotted line, St_0 to St_3 , correspond to the results shown in the top row of Figures 11, 12, and 13. The fifth value, $St + Sh$, corresponds to the final results when shading is turned on. These values have been scaled so that St_0 is equal to one for all triplets. As in the synthetic case, when using stereo alone, the stereo component always improves, but as the recovered surface becomes rougher the shading term degrades dramatically. However, when we turn on the shading component, the overall results improve significantly, even though the stereo component degrades slightly.

6 Summary and Conclusion

In this paper we have presented a surface reconstruction method that uses an object-centered representation (a triangulated mesh) to recover geometry and reflectance properties from multiple images. It allows us to handle self-

occlusions while merging information from several viewpoints, thereby allowing us to eliminate blindspots and making the reconstruction more robust where more than one view is available. The reconstruction process relies on both monocular shading cues and stereoscopic cues. We use these cues to drive an optimization procedure that takes advantage of their respective strengths while eliminating some of their weaknesses.

Specifically, stereo information is very robust in textured regions but potentially unreliable elsewhere. We therefore use it mainly in such areas by weighting the stereo component most strongly for facets of the triangulation that project into textured image areas. The component compares the grey-levels of the points in all of the images for which the projection of a given point on the surface is visible, as determined using a hidden-surface algorithm. This comparison is done for a uniform sampling of

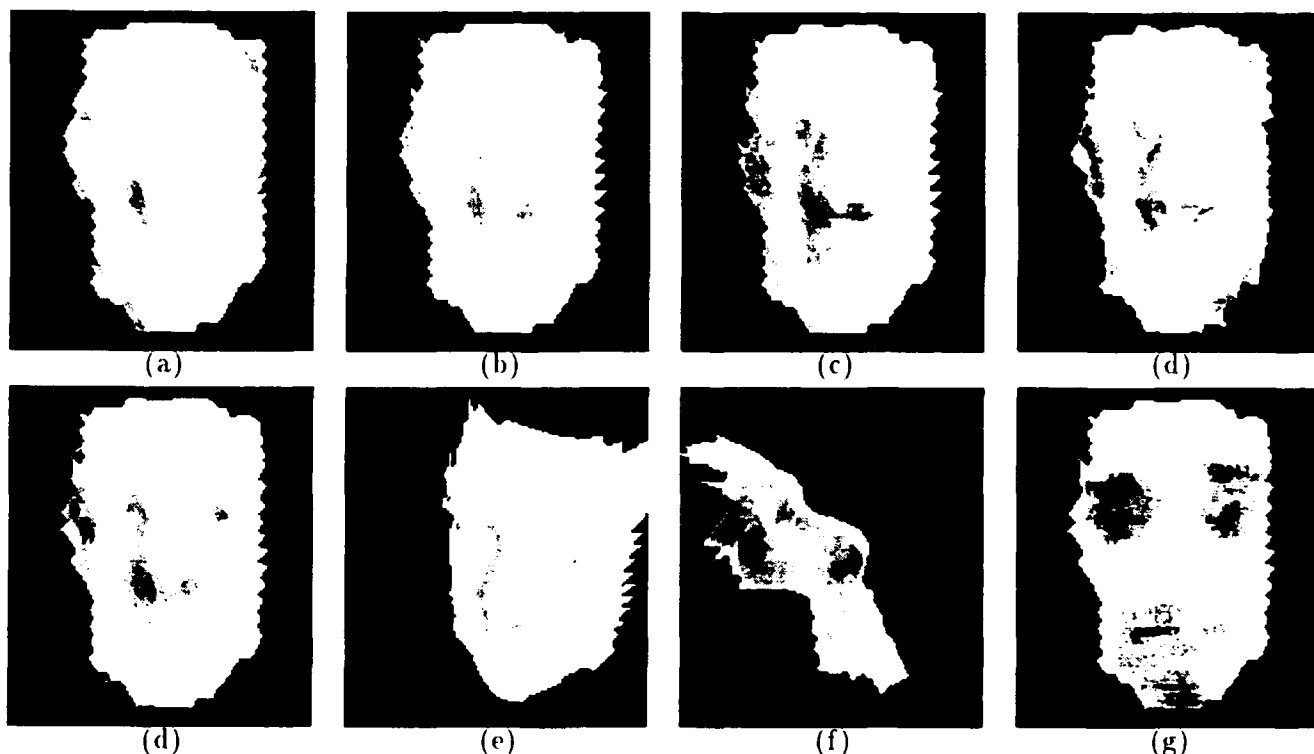


Figure 12: Results for the second triplet of Figure 10 presented in the same fashion as in Figure 11.

the surface. This method allows us to deal with arbitrarily slanted regions and to discount occluded areas of the surface.

On the other hand, shading information is mostly helpful in textureless areas. Thus, we weight the shading component most strongly for facets that project into such areas. The component uses a new method for utilizing shading information that does not need the traditional assumption of constant albedo. Instead, it attempts to minimize the variation in albedo across the surface, and can therefore deal with both constant albedo surfaces and surfaces whose albedo varies slowly. However, it does require the boundary conditions that are provided by the stereo information.

We have developed a weighting scheme that allows our system to use each source of information where it is most appropriate. As a result, for the large class of surfaces that roughly satisfy the Lambertian model, it performs significantly better than if it were using either source of information alone.

Our surface model can be naturally aug-

mented to include specularities, shadows and self-shadows. It can also support more complex topologies, multiple resolutions and the shrinking or growing of the surface of interest, though in this paper we concentrated on a better understanding of the behavior of the objective function. These extensions will be the subject of future work.

Acknowledgments

We wish to thank Hervé Matthieu and Olivier Monga who have provided us with the face images and corresponding calibration data that appear in this paper that have proved extremely valuable to our research effort. We would also like to apologize to the members of the INRIA ROBOTVIS project whose faces we have mercilessly deformed during the development of the algorithms discussed above.

References

- [1] A. L. Abbot and N. Ahuja. Active surface reconstruction by integrating focus, vergence,

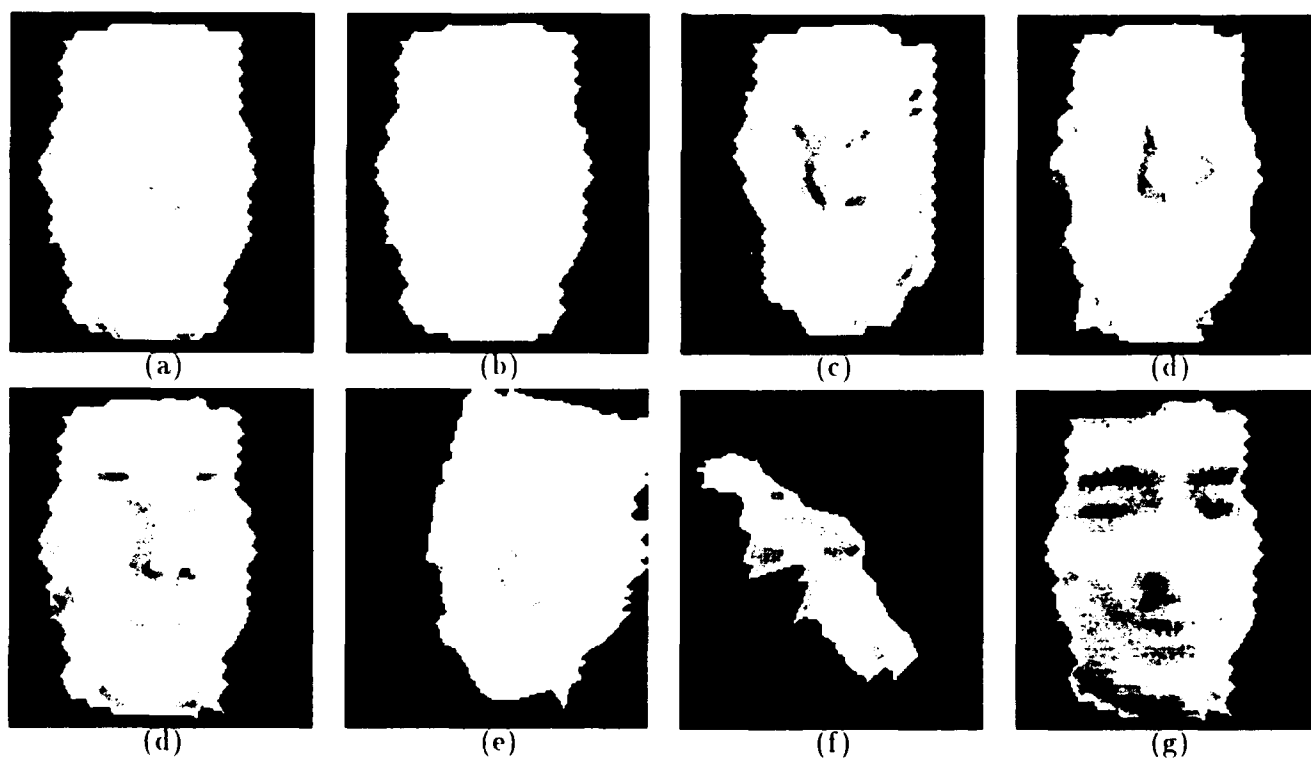


Figure 13: Results for the third triplet of Figure 10 presented in the same fashion as in Figure 11.

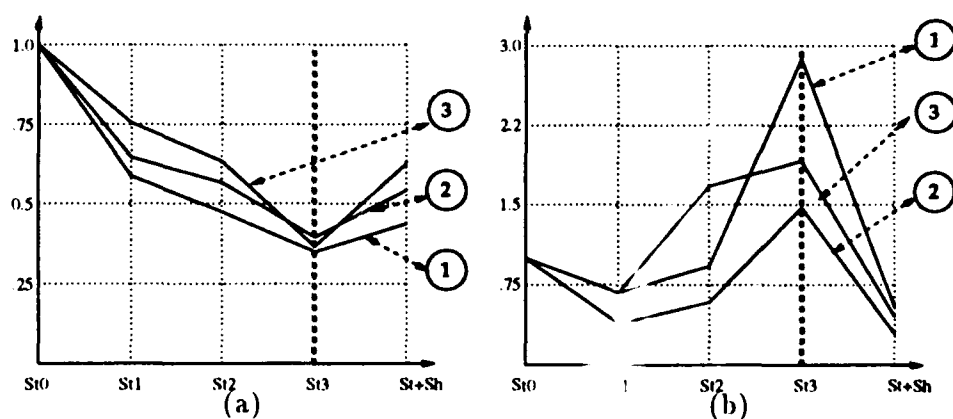


Figure 14: Values of the stereo (a) and shading (b) components of the objective function for the face images. The y axis represents the value of the components and the x axis the various stages of the optimization. From left to right, we first use only stereo and decrease the smoothness and, to the right of the thick dotted line, we turn on the shading term. Each curve is labeled with the number of the corresponding image triplet and all values have been scaled so that the initial ones are equal to 1.0.

stereo, and camera calibration. In *ICCV*, pages 489-492, 1990.

- [2] J. Y. Aloimonos. Unification and integration of visual modules: an extension of the marr paradigm. In *IJCV*, pages 507-551, 1989.

- [3] M. Asada, M. Kimura, Y. Taniguchi, and Y. Shirai. Dynamic integration of height maps into a 3d world representation from range image sequences. *IJCV*, 9(1):31-54, October 1992.

- [4] E. P. Baltsavias. *Multiphoto Geometrically Constrained Matching*. PhD thesis, Institute for Geodesy and Photogrammetry, ETH Zurich, December 1991.
- [5] S. Barnard. Stochastic stereo matching over scale. *Int'l J. Computer Vision*, 3(1):17-32, 1989.
- [6] H. G. Barrow and J. M. Tenenbaum. Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*, pages 3-26. Academic Press, New York, New York, 1978.
- [7] A. Blake, A. Zisserman, and G. Knowles. Surface descriptions from stereo and shading. *Image Vision Comput.*, 3(4):183-191, 1985.
- [8] Y. Choe and R. L. Kashyap. 3-d shape from a shaded and textural surface image. *T-PAMI*, 13:907-919, 1991.
- [9] I. Cohen, L. D. Cohen, and N. Ayache. Introducing new deformable surfaces to segment 3d images. In *CVPR*, pages 738-739, 1991.
- [10] J. E. Cryer, Ping-Sing Tsai, and Mubarak Shah. Combining shape from shading and stereo using human vision model. Technical Report CS-TR-92-25, U. Central Florida, 1992.
- [11] H. Delingette, M. Hebert, and K. Ikeuchi. Shape representation and image segmentation using deformable surfaces. In *CVPR*, pages 467-472, 1991.
- [12] H. Diehl and C. Heipke. Surface reconstruction from data of digital line cameras by means of object based image matching. In *ISPRS*, pages 287-294, Washington D.C., 1992.
- [13] O.D. Faugeras and G. Toscani. The Calibration Problem for Stereo. In *Proceedings of CVPR86, Miami Beach, Florida*, pages 15-20, 1986.
- [14] Frank P. Ferrie, Jean Lagarde, and Peter Whaite. Recovery of volumetric object descriptions from laser rangefinder images. In *European Conference on Computer Vision*, Genoa, Italy, April 1992.
- [15] P. Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 1993. In print, available as INRIA research report 1369.
- [16] P. Fua and Y. G. Leclerc. Model driven edge detection. *Machine Vision and Applications*, 3:45-56, 1990.
- [17] P. Fua and P. Sander. Reconstructing surfaces from unstructured 3d points. In *Proceedings of the 1992 DARPA Image Understanding Workshop*, San Diego, California, January 1992.
- [18] P. Fua and P. Sander. Reconstructing surfaces from unstructured 3d points. In *Image Understanding Workshop*, San Diego, California, January 1992.
- [19] P. V. Fua. Combining stereo and monocular information to compute dense depth maps that preserve depth discontinuities. In *Proceedings of IJCAI*, Sydney, Australia, August 1991.
- [20] W. E. L. Grimson and D. P. Huttenlocher. Introduction to the special issue on interpretation of 3-d scenes. *T-PAMI*, 14(2):97-98, February 1992.
- [21] K. Hartt and M. Carlotto. A method for shape-from-shading using multiple images acquired under different viewing and lighting conditions. In *CVPR*, pages 53-60, 1989.
- [22] C. Heipke. Integration of digital image matching and multi image shape from shading. In *ISPRS*, pages 832-841, Washington D.C., 1992.
- [23] W. Hoff and N. Ahuja. Surfaces from stereo: integrating feature matching, disparity estimation, and contour detection. *T-PAMI*, 11:121-136, 1989.
- [24] B. K. P. Horn. Height and gradient from shading. *Int'l J. Computer Vision*, 5(1):37-75, 1990.
- [25] Y. Hung, D. B. Cooper, and B. Cernuschi-Frias. Asymptotic bayesian surface estimation using an image sequence. *IJCV*, 6(2):105-132, June 1991.
- [26] B. Kaiser, M. Schmolla, and B. P. Wrobel. Application of image pyramid for surface reconstruction with fast vision. In *ISPRS*, page 1, Washington, D.C., 1992.
- [27] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321-331, 1988.
- [28] Y. G. Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision*, 3(1):73-102, 1989.

- [29] Y. G. Leclerc. *The Local Structure of Image Intensity Discontinuities*. PhD thesis, McGill University, Montréal, Québec, Canada, May 1989.
- [30] Y. G. Leclerc and A. F. Bobick. The direct computation of height from shading. In *Proceedings of the 1991 Computer Society Conference on Computer Vision and Pattern Recognition*, Lahaina, Maui, Hawaii, June 1991.
- [31] Y. G. Leclerc and A. F. Bobick. The direct computation of height from shading. In *Proceedings of the 1991 DARPA Image Understanding Workshop*, San Diego, California, January 1992.
- [32] C. E. Liedtke, H. Busch, and R. Koch. Shape adaptation for modelling of 3d objects in natural scenes. In *CVPR*, pages 704-705, 1991.
- [33] D. G. Lowe. Fitting parameterized three-dimensional models to images. *T-PAMI*, 13(441-450), 1991.
- [34] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, Menlo Park, California, second edition, 1984.
- [35] D. Marr. *Vision*. W. H. Freeman, San Francisco, California, 1982.
- [36] A. Pentland. Automatic extraction of deformable part models. *International Journal of Computer Vision*, 4(2):107-126, March 1990.
- [37] A. Pentland and S. Sclaroff. Closed-form solutions for physically based shape modeling and recognition. *T-PAMI*, 13:715-729, 1991.
- [38] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical recipes, the art of scientific computing*. Cambridge U. Press, Cambridge, MA, 1986.
- [39] L. H. Quam. Hierarchical warp stereo. In *Proceedings of the 1984 DARPA Image Understanding Workshop*, pages 149-155, 1984.
- [40] E. M. Stokely and S. Y. Wu. Surface parameterization and curvature measurement of arbitrary 3-d objects: five practical methods. *T-PAMI*, 14(8):833-839, August 1992.
- [41] R. Szeliski. Shape from rotation. In *CVPR*, pages 625-630, 1991.
- [42] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. In *Computer Graphics (SIGGRAPH'92)*, pages 185-194, July 1992.
- [43] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:413-424, 1986.
- [44] D. Terzopoulos. The computation of visible-surface representations. *T-PAMI*, pages 417-438, 1988.
- [45] D. Terzopoulos and D. Metaxas. Dynamic 3d models with local and global deformations: Deformable superquadrics. *T-PAMI*, 13(703-714), 1991.
- [46] D. Terzopoulos and M. Vasilescu. Sampling and reconstruction with adaptive meshes. In *CVPR*, pages 70-75, 1991.
- [47] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-seeking models and 3d object reconstruction. *IJCV*, 1:211-221, 1987.
- [48] C. Tomasi and T. Kanade. The factorization method for the recovery of shape and motion from image streams. In *Proceedings of the 1992 DARPA Image Understanding Workshop*, pages 459-472. DARPA, January 1992.
- [49] B. C. Vemuri and R. Malladi. Deformable models: Canonical parameters for surface representation and multiple view integration. In *CVPR*, pages 724-725, 1991.
- [50] Y. F. Wang and J. F. Wang. Surface reconstruction using deformable models with interior and boundary constraints. *T-PAMI*, 14(5):572-579, May 1992.
- [51] P. Whaite and F. P. Ferrie. From uncertainty to visual exploration. *T-PAMI*, 13(1038-1049), 1991.
- [52] A. W. Witkin, D. Terzopoulos, and M. Kass. Signal matching through scale space. *International Journal of Computer Vision*, 1:133-144, 1987.
- [53] B. P. Wrobel. The evolution of digital photogrammetry from analytical photogrammetry. *Photogrammetric Record*, 13(77):765-776, April 1991.

Provably Convergent Algorithms for Shape from Shading

John Oliensis†

Department of Computer Science
University of Massachusetts at Amherst
Amherst, Massachusetts 01003

Paul Dupuis‡

Box F, Division of Applied Mathematics
Brown University
Providence, Rhode Island 02912 *

Abstract

A new approach to shape from shading is described, based on relating this problem to an "equivalent" optimal control problem. The approach leads naturally to an algorithm for surface reconstruction that is simple, fast, provably convergent, and (under suitable conditions) provably convergent to the correct surface. The theoretical basis of the approach is developed in this paper, extending some earlier partial results. In addition, a new reconstruction algorithm is presented that unlike earlier ones can reconstruct a surface with no a priori information about it.

1 Introduction

We have recently developed a new, practical approach to shape from shading, based on relating this problem to an "equivalent" optimal control problem [Oliensis and Dupuis, 1992; Dupuis and Oliensis, 1992a; Oliensis and Dupuis, 1991; Rouy and Tourin, 1991a; Rouy and Tourin, 1991b; Bichsel and Pentland, 1992]. The approach leads naturally to an algorithm for surface reconstruction that is simple, fast, provably convergent, and (under suitable conditions) provably convergent to the correct surface. In experiments on 200×200 and 128×128 real and synthetic images, convergence took fewer than 15 iterations, and less than 10 seconds on a DECstation 5000 [Oliensis and Dupuis, 1992; Dupuis and Oliensis, 1992a; Bichsel and Pentland, 1992]. Typically, the number of iterations required for reconstruction is expected to be approximately constant independent of the image size. It has also been proven that the reconstruction converges to the correct continuous surface in the limit where the pixel spacing is infinitely small [Dupuis and Oliensis, 1992b; Kushner and Dupuis, 1992]. These results are

to be contrasted with traditional shape from shading algorithms which typically require thousands of iterations, and for which no convergence results are known [Horn and Brooks, 1989].

In this paper we develop the theoretical basis of our approach, extending some earlier partial results. In addition, we describe in Section 4 a new algorithm [Oliensis and Dupuis, 1993] that unlike earlier ones can reconstruct a surface with no a priori information about it. Our previous algorithms, which we discuss in sections 2-3, required a small amount of information about the nature of the surface at singular points—defined as maximally bright image points—and the relative heights of the surface at a subset of these points. ([Rouy and Tourin, 1991a; Rouy and Tourin, 1991b] actually require more data). The algorithm presented in Section 4 computes this information automatically. It is capable of fast, robust reconstruction of a general surface even in the presence of $\pm 10\%$ noise in the intensity.

Specifically, we prove in section 2 that (under appropriate conditions) the surface corresponding to a shaded image has an explicit representation in terms of an optimal control problem. Uniqueness of the surface is an immediate consequence; thus, contrary to previous belief, shape from shading is often a well-posed problem and does not need regularisation. In Section 3, we derive two distinct shape reconstruction algorithms. These are proven to converge for both the Jacobi and significantly faster Gauss-Seidel iteration, and shown to give the same surface reconstruction. It is an advantage of our approach that a range of algorithms can be easily derived. Two algorithms are convenient to work with since one is more efficient computationally while the other has a simpler theoretical interpretation.

2 The Representation Theorem

The purpose of this section is to prove the representation theorem that connects the shape to be reconstructed and a deterministic optimal control problem. The main result, given in (2.3) and The-

*This work was supported by the National Science Foundation under grants IRI-9113690, CDA-8922572† and NSF-DMS-9115762‡, and by a grant from DARPA, via TACOM, contract number DAAE07-91-C-R035†.

orem 2.1, is the explicit representation for the surface corresponding to a shaded image. We consider the idealised problem of shape from shading under the usual assumptions. Note that although we assume Lambertian surface reflectance and illumination from a single direction, our results can be extended easily to any "convex" reflectance function as described below. Under these assumptions, the intensity at an image point $r \equiv (x, y)$ is given by the image irradiance equation

$$I(x, y) = \hat{L} \cdot \hat{n},$$

where \hat{L} is a unit vector in the light source direction, the optical axis is along the $-z$ direction, and \hat{n} is the surface normal at the corresponding surface point. $I(\cdot)$ is assumed to be defined on a bounded open subset \mathcal{D} of \mathbb{R}^2 . Representing the surface by the height function $z(x, y)$, and assuming that $z(\cdot)$ is continuously differentiable (though this is not essential),

$$\hat{n} \equiv \frac{(-\nabla z, 1)}{(1 + \|\nabla z\|^2)^{1/2}}.$$

When the illumination is from a general direction, it is useful to represent the surface by its height $f(\cdot)$ measured along the light direction \hat{L} :

$$f(x, y) = \hat{L} \cdot (x, y, z(x, y)).$$

For simplicity, and without loss of generality, we assume that $L_x = 0$, $L_z > 0$. In terms of $f(\cdot)$, the image irradiance equation can be rewritten after some algebra as $H(r, \nabla f(x)) = 0$, where the Hamiltonian

$$H(r, \alpha) = I(r) (1 + \|\alpha\|^2 - 2\alpha_y L_y)^{1/2} + \alpha_y L_y - 1.$$

Note that $H(r, \alpha)$ is a strictly convex function of α . The fact that the image irradiance equation can be rewritten in terms of a strictly convex H is the essential property used below in the proof of the representation theorem, and also in deriving algorithms. Our results can be extended to essentially any image irradiance equation which can be written in terms of a strictly convex H .

Singular points have been recognised as playing a critical role in constraining the surface corresponding to a shaded image [Oliensis and Dupuis, 1992; Oliensis, 1991b; Saxberg, 1989b; Bruss, 1982], and they are crucial in the discussion here as well. The singular points are those image points where the intensity achieves its maximal brightness $I(\cdot) = 1$. Only at these points is the local surface orientation determined from the intensity alone. Let \mathcal{S} denote the set of singular points in the image.

It is easy to see from the form of $H(r, \alpha)$ that $I = 1$ implies $\alpha = 0$, and therefore $\nabla f = 0$ on the set of singular points \mathcal{S} . Thus \mathcal{S} includes all local maxima and minima of $f(x, y)$. We will focus on those singular points corresponding essentially to the local

minima, and use these in determining the surface from its image. (Alternatively, our results could be derived using the local maxima.) To specify precisely the conditions under which our results hold, we introduce some nonstandard terminology. We say that a set $A \subset \mathbb{R}^2$ is smoothly connected if given any two points r and r' in A there is an absolutely continuous ("smooth") path connecting the two. We will assume that the set of singular points is a finite collection of smoothly connected sets. Then since $\nabla f = 0$ on \mathcal{S} , $f(\cdot)$ is constant over each connected component $\mathcal{S}_C \subset \mathcal{S}$.

We will refer to a connected subset \mathcal{S}_C as a set of local minima if there exists an $\epsilon > 0$ such that $d(r, \mathcal{S}_C) < \epsilon$ implies $f(r) \geq f(r')$ for $r' \in \mathcal{S}_C$, i.e., if the "heights" f of nearby points are no less than the value of f on \mathcal{S}_C . We will refer to a point as a local minimum of f only if it is contained in such a connected subset \mathcal{S}_C . An analogous definition is used for local maxima. A connected subset that is neither a set of local maxima or local minima is called a set of saddle points, even though some of the points in this set may be local minima according to the usual definition (they cannot be strict local minima). Let \mathcal{M} be the set of all the local minima in the above sense.

The Lagrangian corresponding to $H(\cdot)$ is:

$$\begin{aligned} L(r, \beta) &= \sup_{\alpha} [-\alpha \cdot \beta - H(r, \alpha)] \\ &= L_x^2 - L_y \beta_y - L_z (I^2(r) - |\beta_x|^2 - |\beta_y + L_y|^2)^{1/2} \end{aligned} \quad (2.1)$$

if $|\beta_x|^2 + |\beta_y + L_y|^2 \leq I^2(r)$. Otherwise, $L(\cdot) = \infty$. Define $\mathcal{U}(r) \equiv \{\beta : |\beta_x|^2 + |\beta_y + L_y|^2 \leq I^2(r)\}$ to be the domain on which $L(r, \cdot)$ is finite.

The Lagrangian L serves as the running cost in the optimal control problem that provides a representation for the surface, which we now define. Consider an arbitrary path ϕ in the image plane starting at some r , and continuing for a time ρ . More precisely, the path is defined by $\phi(0) = r$, $\dot{\phi} = u(t)$, where the control $u : [0, \infty) \rightarrow \mathbb{R}^2$ is any integrable function. For each such path, we define a cost which is the sum of two terms: 1) the total running cost, given by the integral of the running cost $L(\phi, u(\phi))$ over the path, and 2) a terminal cost, which depends only on the end point of the path, i.e. on $\phi(\rho)$. The control problem is to find the path giving the minimal total cost. The representation theorem states that under appropriate conditions the infimal value of the cost for starting point r is just $f(r)$.

Assume we are given an upper bound B for $\{f(r) : r \in \mathcal{D}\}$. Then define the terminal cost

$$g(r) = \begin{cases} f(r) & \text{for } r \in \mathcal{M} \\ B & \text{for } r \notin \mathcal{M} \end{cases} \quad (2.2)$$

The terminal cost imposes the large penalty B on any path terminating at a point $r \notin \mathcal{M}$. Finally, the

total cost is the sum of the running and terminal costs

$$V(r) = \inf \left[\int_0^{\rho \wedge \tau} L(\phi(s), u(s)) ds + g(\phi(\rho \wedge \tau)) \right], \quad (2.3)$$

where $\rho \wedge \tau$ denotes $\min(\rho, \tau)$, $\tau = \inf\{t : \phi(t) \in \partial\mathcal{D} \cup \mathcal{M}\}$ and the infimum is over all paths ϕ and stopping times $\rho \in [0, \infty)$. Thus, V is the "minimal" cost over all finite time paths, where the path terminates either at time ρ determined by the controller, or else at the first time that the path exits \mathcal{D} or enters \mathcal{M} . We want to show that $f(r) = V(r)$.

Preliminaries. For any H , including nonconvex H , it is easy to show that the definition (2.1) implies that the running cost $L(r, \cdot)$ is convex on $\mathcal{U}(r)$. In our case, it is strictly convex. Moreover, a direct calculation shows that L is always nonnegative, and $L(r, \beta) = 0$ only for $r \in \mathcal{S}$ and $\beta = 0$. Also, since $H(r, \cdot)$ is strictly convex, it follows by standard arguments that

$$H(r, \alpha) = \sup_{\beta \in \mathcal{U}(r)} [-\alpha \cdot \beta - L(r, \beta)], \quad (2.4)$$

and for each $\alpha \in \mathbb{R}^2$ there exists a unique vector $u(r, \alpha)$ such that

$$H(r, \alpha) = -\alpha \cdot u(r, \alpha) - L(r, u(r, \alpha)).$$

Define $\bar{u}(r)$ for $r \in \mathcal{D}$ by

$$0 = H(r, \nabla f(r)) = -\nabla f(r) \cdot \bar{u}(r) - L(r, \bar{u}(r)). \quad (2.5)$$

From (2.4), $\bar{u}(r)$ is given by

$$\nabla_\alpha H(r, \alpha)|_{\nabla f(r)} = -\bar{u}(r).$$

If (as we assume) $\nabla f(r)$ is continuous, then the fact that $H(r, \cdot)$ is C^1 implies $\bar{u}(r)$ is continuous on \mathcal{D} . An explicit calculation shows that $\bar{u}(r)$ is proportional to the projection in the (x, y) plane of the steepest descent direction on the surface [Oliensis, 1991a], where "steepest descent" is defined with respect to the light direction \hat{L} , rather than the vertical direction $(0, 0, 1)$.

We consider subsets \mathcal{G} of \mathcal{D} satisfying the following assumption.

A2.1 Assume that \mathcal{S} consists of a finite collection of disjoint, compact, smoothly connected sets, and that $\nabla f(\cdot)$ is continuous on the closure of \mathcal{D} . Let $\mathcal{G} \subset \mathcal{D}$ be a compact set, and assume \mathcal{G} is of the form $\mathcal{G} = \bigcap_{j=1}^J \mathcal{G}_j$, $J < \infty$, where each \mathcal{G}_j has a continuously differentiable boundary. Let \mathcal{M} be the set of local minima of $f(\cdot)$ inside \mathcal{G} . Then we assume that the value of $f(\cdot)$ is known at all points in \mathcal{M} . Let \bar{u} denote the "steepest descent" direction given by (2.5) above. Define $n_j(r)$ to be the inward (with respect to \mathcal{G}) normal to $\partial\mathcal{G}_j$ at r . Then we also assume that $\bar{u}(r) \cdot n_j(r) > 0$ for all $r \in \partial\mathcal{G} \cap \partial\mathcal{G}_j$, $j = 1, \dots, J$.

As noted above, the minimising trajectories for the control problems we consider are the two dimensional projections of the paths of steepest descent on the surface. The assumption on \mathcal{G} above states that the steepest descent direction is never directed out of \mathcal{G} , and thus guarantees that any minimising trajectory that starts in \mathcal{G} stays in \mathcal{G} . When this assumption is violated for some point $r \in \mathcal{D}$, i.e. when the steepest descent trajectory starting at r exits \mathcal{D} , then $f(r)$ cannot be represented as the minimal cost $V(r)$. However, if a steepest ascent path starting at r does remain in \mathcal{D} , then $f(r)$ can be computed in terms of a maximum cost for an analogous optimal control problem. If neither of these possibilities hold, then the surface at r is not well determined. In general, this is expected to be the case only for small sections of the image near the image boundary [Oliensis, 1991a].

Theorem 2.1 Assume A2.1, and that B is an upper bound for $f(\cdot)$ on \mathcal{G} . Define $L(\cdot, \cdot)$ by (2.1), $g(\cdot)$ by (2.2), and $V(r)$ by (2.3). Then $V(r) = f(r)$ for all $r \in \mathcal{G}$.

Proof. We first show that $V(r) \geq f(r)$. Let $u(\cdot)$ be any admissible control and define

$$\phi(t) = r + \int_0^t u(s) ds, \quad \tau = \inf\{t : \phi(t) \in \partial\mathcal{D} \cap \mathcal{M}\}. \quad (2.6)$$

Since L is the Legendre transform of H and since $H(r, \nabla f(r)) = 0$ for $r \in \mathcal{G}$,

$$0 \geq -\nabla f(r) \cdot \beta - L(r, \beta)$$

for all $\beta \in \mathbb{R}^2$, and in particular

$$-\nabla f(\phi(t)) \cdot u(t) \leq L(\phi(t), u(t))$$

for $t \in [0, \rho \wedge \tau]$. This implies that

$$\begin{aligned} -f(\phi(\rho \wedge \tau)) + f(r) &= -\int_0^{\rho \wedge \tau} \nabla f(\phi(t)) \cdot u(t) dt \\ &\leq \int_0^{\rho \wedge \tau} L(\phi(t), u(t)) dt, \end{aligned}$$

and thus

$$\int_0^{\rho \wedge \tau} L(\phi(t), u(t)) dt + f(\phi(\rho \wedge \tau)) \geq f(r).$$

Since $g(\phi(\rho \wedge \tau)) \geq f(\phi(\rho \wedge \tau))$, we obtain $V(r) \geq f(r)$.

Next we show $V(r) \leq f(r)$. In order to do so we will verify that for each $\epsilon > 0$ there exists a control $u(\cdot)$ such that for ϕ and τ defined by (2.6) we have

$$\int_0^\tau L(\phi(t), u(t)) dt + g(\phi(\tau)) \leq f(r) + \epsilon. \quad (2.7)$$

Recall that $\mathcal{U}(r) = \{(\beta_x, \beta_y) : |\beta_x|^2 + |\beta_y|^2 + L_y \leq 1\}$ and $L(r, 0) = 0$ for $r \in \mathcal{S}$. Let \mathcal{S}_C be a maximal smoothly connected component of \mathcal{S} . For any two

points r, r' in S_C , there exists a path $\phi(t)$ and a time $t^* < \infty$ such that $\phi(t) \in S_C$ for $t \in [0, t^*]$ and $r' = \phi(t^*)$. Write $\phi(t) = r + \int_0^t u(s)ds$ in terms of the control $u(t)$. Define a new control $u_\lambda(t) \equiv \lambda u(t\lambda)$, where $\lambda > 0$ is a constant, and let $\phi_\lambda(t) = \phi(t\lambda)$ be the corresponding path. Since

$$\frac{L(r, u)}{\|u\|} \rightarrow 0 \text{ as } \|u\| \rightarrow 0,$$

for r such that $I(r) = 1$, we can choose λ such that

$$\int_0^{\lambda t^*} L(\phi_\lambda(t), u_\lambda(t))dt = \int_0^{t^*} \frac{L(\phi(t), \lambda u(t))}{\lambda} dt \leq \frac{\epsilon}{3}.$$

Further, since $|L_y| < 1$, there exists $\alpha > 0$ such that for any component S_C as above, and r such that $d(r, S_C) \leq \alpha$, we have the following. Let r' be the point in S_C closest to r . Then there exists a time $t_a \in [0, \infty)$, constant control $u(\cdot) = (r' - r)/t_a$ and corresponding path $\phi(t) = r + \int_0^t u(s)ds$, such that $\phi(t_a) = r'$ and $\int_0^{t_a} L(\phi(t), u(t))dt \leq \epsilon/3$. Finally, this shows that for any S_C , and r, r' such that $d(r, S_C) \leq \alpha$ and $d(r', S_C) \leq \alpha$, there exists a control $\bar{u}_{rr'}(t)$ and time $\sigma_{rr'} \in [0, \infty)$ such that for the corresponding path $\phi_{rr'}(t)$ we have $\phi_{rr'}(0) = r$, $\phi_{rr'}(\sigma_{rr'}) = r'$, and

$$\int_0^{\sigma_{rr'}} L(\phi_{rr'}(t), \bar{u}_{rr'}(t))dt \leq \epsilon.$$

Since f is constant on S_C , then by choosing $\alpha > 0$ smaller if need be we can also assume that $|f(r) - f(r')| \leq \epsilon$.

We now construct the control that satisfies (2.7). If r is a local minimum then we simply take $\tau = 0$ and are done.

There are then two remaining cases: (1) r is contained in some S_C with $S_C \cap \mathcal{M} = \emptyset$, or (2) $r \notin S$. If case (1) holds then $S_C \cap \mathcal{M} = \emptyset$ implies the existence of a point r' such that $f(r') < f(r)$ and $d(r', S_C) \leq \alpha$. Since A2.1 implies $S \subset \mathcal{G}^0$, we can assume that $r' \in \mathcal{G}$. In this case we will set $u(t) = \bar{u}_{rr'}(t)$ for $t \in [0, \sigma_{rr'})$.

Next consider the definition of the control for $t \geq \sigma_{rr'}$. For $c > 0$ let $b = \inf\{L(r, u) : r \in \mathcal{D}, d(r, S) > c, u \in \mathbb{R}^2\}$. The continuity of $I(\cdot)$ and the fact that $I(r) < 1$ for $r \notin S$ implies $b > 0$. Consider any solution (there may be more than one) to

$$\dot{\phi}(t) = \bar{u}(\phi(t)), \phi(0) = r'. \quad (2.8)$$

According to (2.5), for any t such that $\phi(t) \in \mathcal{G} \setminus S$ and $d(\phi(t), S) > c$

$$\begin{aligned} \frac{d}{dt} f(\phi(t)) &= \nabla f(\phi(t)) \cdot \bar{u}(\phi(t)) \\ &= -L(\phi(t), \bar{u}(\phi(t))) \\ &\leq -b. \end{aligned} \quad (2.9)$$

A2.1 implies $\phi(t)$ cannot exit \mathcal{G} . Thus, since $f(r)$ is bounded on \mathcal{G} , (2.9) implies that $\phi(t)$ must enter the set $\{r : d(r, S) \leq c\}$ in finite time, for any c . If $\phi(t) \in S$ for some $t < \infty$ we define $\eta_{r'} = \inf\{t : \phi(t) \in S\}$ and $w = \phi(\eta_{r'})$. Otherwise, let t_i be any sequence tending to ∞ as $i \rightarrow \infty$. Since \mathcal{G} is compact we can extract a subsequence (again labeled by i) such that $\phi(t_i) \rightarrow v$ for some $v \in S$. Let \bar{i} be large enough that $\|\phi(t_{\bar{i}}) - v\| \leq \alpha$. Since $f(\phi(t_{\bar{i}})) \downarrow f(v)$, we have $f(\phi(t_{\bar{i}})) > f(v)$. For this case we define $\eta_{r'} = t_{\bar{i}}$ and $w = \phi(\eta_{r'})$.

Integrating (2.9) gives

$$f(r') - f(w) = \int_0^{\eta_{r'}} L(\phi(t), \bar{u}(\phi(t)))dt.$$

We then define the control $u(t)$ to be used for $t \in [\sigma_{rr'}, \sigma_{rr'} + \eta_{r'})$ to be $\bar{u}(\phi(t - \sigma_{rr'}))$.

We now consider the point w . We first examine the case in which the solution to (2.8) does not enter S in finite time. Since $\|w - v\| \leq \alpha$, $\bar{u}_{wv}(t)$ gives a control such that the application of this control moves $\phi(\cdot)$ from w to v with accumulated running cost less than or equal to ϵ . We define $u(t) = \bar{u}_{wv}(t - (\eta_{r'} + \sigma_{rr'}))$ for $t \in [\sigma_{rr'} + \eta_{r'}, \sigma_{rr'} + \eta_{r'} + \sigma_{wv})$. If the solution to (2.8) reached S in finite time we define $w = v$ and $\sigma_{wv} = 0$. Let $\sigma = \sigma_{rr'} + \eta_{r'} + \sigma_{wv}$.

Let us summarise the results of this construction. Given any point $r \in S$ that is not a local minimum we have constructed a piecewise continuous control $u(\cdot)$ and $\sigma < \infty$ such that if $\phi(t) = r + \int_0^t u(s)ds$, then

$$\begin{aligned} f(r) - f(\phi(\sigma)) &= f(r) - f(r') + f(r') \\ &\quad - f(w) + f(w) - f(v) \\ &\geq \int_{\sigma_{rr'}}^{\sigma_{rr'} + \eta_{r'}} L(\phi(t), u(t))dt \\ &\geq -2\epsilon + \int_0^{\sigma} L(\phi(t), u(t))dt. \end{aligned}$$

We have also shown that $f(r) > f(v) = f(\phi(\sigma))$, $\phi(\sigma) \in S$. Thus, either the component S_C containing $\phi(\sigma)$ satisfies $S_C \cap \mathcal{M} \neq \emptyset$, and we are done, or we are back into case (1) above, and can repeat the procedure. Let K be the number of disjoint compact connected sets that comprise S . Then the strict decrease $f(r) > f(\phi(\sigma))$ and the fact that $f(\cdot)$ is constant on each S_C imply the procedure can be repeated no more than K times before reaching some S_C containing a point from \mathcal{M} . If case (2) holds we can use the same procedure, save that the very first step is omitted. Thus, in general, we have exhibited a control $u(\cdot)$ such that

$$\int_0^{\tau} L(\phi(t), u(t))dt + g(\phi(\tau)) \leq f(r) + (2K + 1)\epsilon.$$

Since $\epsilon > 0$ is arbitrary, the theorem is proved. ■

3 Shape Reconstruction Algorithms

In this section, we describe how algorithms for shape reconstruction can be derived from control representations such as that given in the previous section. It is important to note that many different algorithms can be derived, corresponding to the many possibilities for rewriting the image irradiance equation in terms of a Hamiltonian. Each choice for the Hamiltonian leads in general to a different control representation (at least formally), and a different algorithm. Nevertheless, the different algorithms compute the same surface approximation from the image. Thus, for example, an algorithm can be generated from the Hamiltonian of the previous section, which we henceforth denote by $H^{(1)}$. Another possibility, used previously in [Oliensis and Dupuis, 1992], is to write the image irradiance equation in the form $H^{(2)}(r, \nabla f(r)) = 0$, with $H^{(2)}(r, \alpha)$ given by

$$\frac{1}{2} [I^2 \alpha_x^2 + v \alpha_y^2 + 2(1 - I^2) L_y \alpha_y - (1 - I^2)], \quad (3.10)$$

where $v(r) = I^2(r) - L_y^2$. Note that when $v(r) < 0$ $H^{(2)}(r, \alpha)$ is not a convex function of α . Nevertheless, an algorithm can be derived from this form of the Hamiltonian, which, although it differs from the algorithm generated from $H^{(1)}$, reconstructs the same surface approximation.

The algorithms are derived using a discrete approximation of the continuous control representation. In this discrete control problem, the object is to minimize the cost over all discrete paths on the grid of pixels. A difficulty in doing this is that a discrete trajectory, where at each time step the path jumps between neighboring pixels, is usually a poor approximation to a continuous trajectory. In order to better approximate a continuous trajectory on a discrete grid, an element of randomness is introduced. Thus the continuous optimal control problem is approximated by a discrete stochastic optimal control problem, and the cost of the continuous problem is approximated by the expectation of the cost for the discrete problem. Note that the algorithms themselves are deterministic, even though the discrete control problem involves a stochastic rather than deterministic process.

Thus, given a control u , we define the probabilities for the path to jump to neighboring pixels so that on average the discrete motion approximates the continuous motion $\dot{\phi} = u$. Let $p(r, r'|u)$ denote the transition probability for the path to move from r to a 4-nearest neighbor site r' in the current time step. We define

$$p(r, r + \text{sign}(u_x)(1, 0)|u) = \frac{|u_x|}{|u_x| + |u_y|} \quad (3.11)$$

$$p(r, r + \text{sign}(u_y)(0, 1)|u) = \frac{|u_y|}{|u_x| + |u_y|}, \quad (3.12)$$

with all other probabilities zero. We also define the size of the time step to be $\Delta t(u) \equiv 1/(|u_x| + |u_y|)$. With this definition, and assuming for example $u_x, u_y > 0$, the average motion is

$$\frac{(1, 0)u_x + (0, 1)u_y}{|u_x| + |u_y|} = u \Delta t(u),$$

which approximates the continuous motion. This definition actually makes sense only when $u \neq 0$. For $u = 0$, we define $p(r, r|0) = 1$, and $\Delta t(0) = 1$.

For a given sequence of controls $\{u_i\}$, let $\{\xi_i : \xi_0 = r\}$ denote the path starting at r which evolves at each time step i as determined by the control sequence $\{u_i\}$ and the transition probabilities. Then for the representation and Lagrangian (now denoted $L^{(1)}$) of the previous section, the infimal cost $V^{(1)}(r)$ of the approximating stochastic control problem is given by

$$\inf E_x \left[\sum_{i=0}^{(N \wedge M)-1} L^{(1)}(\xi_i, u_i) \Delta t(u_i) + g(\xi_{(N \wedge M)}) \right],$$

where $N = \inf\{i : \xi_i \notin \mathcal{D} \text{ or } \xi_i \in \mathcal{M}\}$, and the minimisation is over all control sequences $\{u_i\}$ and stopping times M . See [Dupuis and Oliensis, 1992b] for the description of the classes of allowed controls and stopping times. E_x denotes the expectation. Thus, $V^{(1)}$ is the minimum of the expectation of the cost over all finite length control sequences, where the path terminates either at discrete time M chosen by the controller, or else at the first time that the path exits \mathcal{D} or enters \mathcal{M} .

Suppose that instead of considering paths of arbitrary length, we consider paths continuing for at most n time steps. The infimal cost $V_n^{(1)}(r)$ is

$$\inf E_x \left[\sum_{i=0}^{(N \wedge M \wedge n)-1} L^{(1)}(\xi_i, u_i) \Delta t(u_i) + g(\xi_{(N \wedge M \wedge n)}) \right]. \quad (3.13)$$

Then $V_n^{(1)}(r)$ is clearly nonincreasing in n and $V_n^{(1)}(r) \downarrow V^{(1)}(r)$ as $n \rightarrow \infty$. For $a \in \{1, 2\}$, define $W^{(a)}(r, u, V_n^{(a)})$

$$\equiv L^{(a)}(r, u) \Delta t(u) + \sum_{r'} p(r, r'|u) V_n^{(a)}(r'), \quad (3.14)$$

where the sum is over 4-nearest neighbors of r . As discussed in [Oliensis and Dupuis, 1992], it follows from the principle of dynamic programming that $V_n^{(1)}(r)$ and $V_{n+1}^{(1)}(r)$ are related by

$$V_{n+1}^{(1)}(r) \equiv \min \left[\inf_{u \in \mathbb{R}^2} W^{(1)}(r, u, V_n^{(1)}), g(r) \right]. \quad (3.15)$$

Clearly, we also have the initial condition $V_0^{(1)}(r) = g(r)$. This, together with the recursive equation

(3.15), gives an algorithm which converges monotonically down to V .

For the second control problem, we get a similar algorithm. As before, $V_0^{(2)}(r) = g(r)$. If $v(r) \geq 0$,

$$V_{n+1}^{(2)}(r) = \min \left[\inf_{u \in \mathbb{R}^2} W^{(2)}(r, u, V_n^{(2)}), g(r) \right],$$

else if $v(r) < 0$

$$V_{n+1}^{(2)}(r) = \min \left[\sup_{u, L_r < 0} \inf_{u \in \mathbb{R}} W^{(2)}(r, u, V_n^{(2)}), g(r) \right]. \quad (3.16)$$

The Lagrangian $L^{(2)}$ is derived from an equation analogous to (2.1): $L^{(2)}(r, \beta)$

$$\begin{aligned} &= \sup_{\alpha} [-\alpha \cdot \beta - H^{(2)}(r, \alpha)] && \text{if } v(r) > 0 \\ &= \inf_{\alpha} \sup_{\alpha} [-\alpha \cdot \beta - H^{(2)}(r, \alpha)] && \text{if } v(r) < 0. \end{aligned}$$

(The case $v(r) = 0$ is given by the appropriate limit as $v(r) \rightarrow 0$ from either direction.) The difference from the previous algorithm is due to the nonconvexity in the Hamiltonian for image regions where $v(r) < 0$. For more detail, and experimental results obtained with the second algorithm above, consult [Dupuis and Oliensis, 1992b; Oliensis and Dupuis, 1992].

The algorithms described above are of the Jacobi type, with the surface updated everywhere in parallel at each iteration. The algorithms can also be shown to converge if implemented via Gauss-Seidel, with updated surface estimates used as soon as they are available. In fact, we show below that the Gauss-Seidel algorithms converge for any sequence of pixel updates, as long as each site is updated a sufficient number of times. For example, it is possible to change the direction of the sweep across the image after each pass [Bichsel and Pentland, 1992]. Our experiments show that this produces a significant speedup, changing the number of iterations required for convergence from order N to order 1 with a small constant, where N is the linear dimension of the image.

Proposition 3.1 Consider either of the recursive algorithms derived in (3.15) or (3.16). Let an initial condition $V_0^{(a)}$, where $a \in \{1, 2\}$, be given and define the sequence $\{V_i^{(a)}, i \in \mathbb{N}\}$ according to either the Jacobi iteration [e.g. (3.15)] or the Gauss-Seidel iteration, where the pixel sites are updated in an arbitrary sequence. Assume that $V_0^{(a)}(r) \geq g(r)$ for all $r \in \mathcal{D}$. Then the following conclusions hold.

1. For each $r \in \mathcal{D}$, $V_i^{(a)}(r)$ is nonincreasing in i and bounded from below. Define $V^{(a)}(r) = \lim_{i \rightarrow \infty} V_i^{(a)}(r)$. Then the function $V^{(a)}(\cdot)$ is a fixed point of (3.15) (or (3.16) if appropriate).

2. The function $V^{(a)}(\cdot)$ can be uniquely characterized as the largest fixed point of (3.15) (or (3.16) if appropriate) that satisfies $V^{(a)}(r) \leq V_0^{(a)}(r)$ for all $r \in \mathcal{D}$.

Remark. In using the algorithm, we always take $V_0^{(a)}(r) = g(r)$, where $g(r)$ is defined by (2.2). It is proven in [Dupuis and Oliensis, 1992b] that if B in (2.2) is an upper bound for $f(r)$, then the surface reconstructed by the algorithms above converges to the correct surface as the pixel spacing goes to zero. Thus, the correct surface approximation is obtained by taking the largest of all the fixed points of the iterations in (3.15) or (3.16).

Proof. For each fixed $r \in \mathcal{D}$, any of the Jacobi and Gauss-Seidel iterations we have defined may be written in one of the following forms. For $a \in \{1, 2\}$, define $W_{\text{old}}^{(a)}(r, u, V_{\text{old}}^{(a)})$

$$\equiv c(r, u) + \sum_{r'} p(r, r' | u) V_{\text{old}}^{(a)}(r'),$$

where $c(r, u)$ denotes the running cost for the given algorithm, and $V_{\text{old}}^{(a)}(\cdot)$ represents the result of previous updates of the algorithm. Then the result $V_{\text{new}}^{(a)}(r)$ after a new update is given either by

$$\min \left[\inf_u \left(W_{\text{old}}^{(a)}(r, u, V_{\text{old}}^{(a)}) \right), g(r) \right], \quad (3.17)$$

or

$$\min \left[\sup_{u, L_r < 0} \inf_u \left(W_{\text{old}}^{(a)}(r, u, V_{\text{old}}^{(a)}) \right), g(r) \right]. \quad (3.18)$$

Note that for both (3.17) and (3.18) the right hand sides are monotonically nondecreasing in $V_{\text{old}}^{(a)}(\cdot)$ if we use the partial ordering of real valued functions on \mathcal{D} defined by $w_1(\cdot) \leq w_2(\cdot)$ whenever $w_1(r) \leq w_2(r)$ for all $r \in \mathcal{D}$.

For either the Jacobi or Gauss-Seidel iteration, the first update at the site r will result in $V_{\text{new}}^{(a)}(r) \leq V_0^{(a)}(r)$, since $V_0^{(a)}(\cdot) \geq g(\cdot)$. Next, consider any subsequent update of the site r . By induction, we can assume that all updates are nonincreasing up to this time. $V_{\text{new}}^{(a)}(r)$ depends on $V_{\text{old}}^{(a)}(r' \neq r)$, which by the induction assumption is everywhere less than or equal to its value at the previous update at r . Since $V_{\text{new}}^{(a)}(r)$ depends monotonically on $V_{\text{old}}^{(a)}(r')$, it follows that $V_{\text{new}}^{(a)}(r)$ also satisfies $V_{\text{new}}^{(a)}(r) \leq V_{\text{old}}^{(a)}(r)$ for the current iteration. This establishes the monotonicity of part 1.

Since for Control Problem 1 the running costs c are nonnegative, $V^{(1)}$ is bounded from below, and the

monotonicity proved above establishes the existence of $V^{(1)}(r) = \lim_{i \rightarrow \infty} V_i^{(1)}(r)$. This is also the case for Control Problem 2 when $v(r) \geq 0$. When $v(r) < 0$ in Control Problem 2, we first consider (3.18) assuming $V_{\text{old}}^{(2)} = 0$. A simple calculation shows that

$$\sup_{u_y L_y < 0} \inf_{u_x} (L^{(2)}(r, u)) > 0,$$

and therefore

$$\sup_{u_y L_y < 0} \inf_{u_x} (L^{(2)}(r, u) \Delta t(u)) \geq 0.$$

Since the probabilities are nonnegative, and $\sum_{r'} p(r, r'|u) = 1$, this implies $\min_{r'} V_{\text{old}}^{(2)}(r') \leq$

$$\sup_{u_y L_y < 0} \inf_{u_x} \left(L^{(2)}(x, u) \Delta t + \sum_{r'} p(r, r'|u) V_{\text{old}}^{(2)}(r') \right).$$

Thus for Control Problem 2 and $v(r) < 0$, $V_n^{(2)}$ is bounded from below by $\min_r V_0^{(2)}(r)$. This gives part 1 of the proposition.

We next turn to part 2. Let $\bar{V}^{(a)}$ be any fixed point of (3.15) or (3.16) that satisfies $\bar{V}^{(a)}(r) \leq V_0^{(a)}(r)$ for all $r \in \mathcal{D}$. An argument very similar to the one used to prove part 1 shows that

$$\bar{V}^{(a)}(r) \leq V_i^{(a)}(r) \Rightarrow \bar{V}^{(a)}(r) \leq V_{i+1}^{(a)}(r).$$

Therefore by induction $\bar{V}^{(a)}(r) \leq V^{(a)}(r)$ for all $r \in \mathcal{D}$. ■

Finally, we sketch the proof that the algorithms of (3.15) and (3.16) converge to the same fixed point. For a complete proof, see [Dupuis and Oliensis, 1992b]. The same argument generalises to show that a wide range of algorithms corresponding to different choices of the Hamiltonian all converge to the same fixed point.

For specificity, consider Control Problem 1, and let $w(x)$ be a fixed point of the corresponding iteration (3.15). We will only consider the case $r \notin \mathcal{S}$ and $w(r) < g(r)$; the other cases can be handled similarly. Then

$$0 = \inf_{u \in \mathbb{R}^2} [L^{(1)}(r, u) \Delta t(u) +$$

$$\sum_{r'} p(r, r'|u) (w(r') - w(r))]$$

Substituting the transition probabilities from (3.11), (3.12) gives

$$0 = \min \left[\inf_{u \neq 0} \frac{L^{(1)}(r, u) + u \cdot (\nabla w)^u(r)}{|u_x| + |u_y|}, L^{(1)}(r, 0) \right] \quad (3.19)$$

where the second expression on the right hand side corresponds to $u = 0$, and $(\nabla w)^u$ is a forward or backward discrete derivative depending on u :

$$(\nabla w)_x^u \equiv \begin{cases} w(r + (1, 0)) - w(r) & \text{if } u_x \geq 0 \\ w(r) - w(r - (1, 0)) & \text{if } u_x < 0, \end{cases}$$

$$(\nabla w)_y^u \equiv \begin{cases} w(r + (0, 1)) - w(r) & \text{if } u_y \geq 0 \\ w(r) - w(r - (0, 1)) & \text{if } u_y < 0. \end{cases}$$

Since we assume $r \notin \mathcal{S}$, $L^{(1)}(r, \cdot) > 0$ and only the first expression on the right hand side of (3.19) need be considered. Then (3.19) is equivalent to

$$0 = \min_u [L^{(1)}(r, u) + u \cdot (\nabla w)^u], \quad (3.20)$$

where we have used the fact that $L^{(1)}$ has a strictly positive lower bound for $r \notin \mathcal{S}$, and that it is finite only on a bounded domain. Also, the minimisation has been extended to $u = 0$ using the continuity of the bracketed expression in u .

Apart from the dependence of $(\nabla w)^u$ on u , eq. 3.20 is the Legendre transform

$$H^{(1)}(x, \alpha) = - \min_u [L^{(1)}(x, u) + u \cdot \alpha]. \quad (3.21)$$

Therefore we have established a connection between the conditions for $w(r)$ to be a fixed point and the equation $H^{(1)}(r, (\nabla w)^u) = 0$, which is a discrete version of the original image irradiance equation. The plan of the proof is to relate in this way the fixed point conditions for any algorithm back to the image irradiance equation. Since all algorithms are derived from this equation in the first place, this will allow us to prove that the fixed point conditions are the same for all algorithms.

For simplicity, we consider only points r for which $v(r) > 0$. A more complete discussion can be found in [Dupuis and Oliensis, 1992b]. Then $H^{(1)}(r, \alpha) = 0$ if and only if $H^{(2)}(r, \alpha) = 0$, since both correspond to the image irradiance equation. Moreover, since both Hamiltonians are convex in α for $v(r) > 0$, $H^{(1)}(r, \alpha) > (<) 0$ if and only if $H^{(2)}(r, \alpha) > (<) 0$, respectively.

By a similar argument to that above, it can be shown that the algorithm for Control Problem 2 has a fixed point at a point $r : v(r) > 0$ if and only if

$$0 = \min_u [L^{(2)}(r, u) + u \cdot (\nabla w)^u]. \quad (3.22)$$

Divide the u plane into four quadrants Q_i where each quadrant is characterised by constant values of $\text{sign}(u_x)$ and $\text{sign}(u_y)$. Then (3.20) and (3.22) can be rewritten as

$$0 = \min_{i=1,2,3,4} \left[\min_{u \in Q_i} (L^{(a)}(r, u) + u \cdot (\nabla w)^i) \right],$$

where $a \in \{1, 2\}$, and $(\nabla w)^i$ is the appropriate choice of the discrete derivative for the given quadrant. Without loss of generality consider the quadrant Q_1 where $u_x, u_y \geq 0$ and $(\nabla w)^1(r) = (w(r + (1, 0)) - w(r), w(r + (0, 1)) - w(r))$. From a simple argument using the convexity of the $L^{(a)}$, it can be proven (see for example [Rockafellar, 1970]) that

$$\min_{u_x, u_y \geq 0} [L^{(a)}(r, u) + u \cdot \alpha] = - \min_{\alpha_x^*, \alpha_y^* \geq 0} H^{(a)}(\alpha - \alpha^*),$$

where $a \in \{1, 2\}$. Then, since the signs of $H^{(1)}$ and $H^{(2)}$ correspond,

$$\min_{u \in Q_1} [L^{(1)}(r, u) + u \cdot (\nabla w)^1] \left\{ \begin{array}{l} > \\ = \end{array} \right\} 0$$

if and only if, respectively,

$$\min_{u \in Q_1} [L^{(2)}(r, u) + u \cdot (\nabla w)^1] \left\{ \begin{array}{l} > \\ = \end{array} \right\} 0.$$

Since this is true for each quadrant, $w(r)$ is a fixed point for Control Problem 1 if and only if it is one for Control Problem 2. ■

4 A General Shape Reconstruction Algorithm

The algorithms above are in a sense local. They use the fact that a singular point—defined as a maximally bright image point—uniquely determines the surface within some image neighborhood [Bruss, 1982; Saxberg, 1989a; Saxberg, 1989b; Oliensis, 1991b]. More precisely, this is true for singular points at which the surface height has a local minimum or local maximum. The algorithms (in the appropriate minimum- or maximum-based version) are guaranteed to give the correct surface up to an overall translation in depth near each such singular point.

The surface over the whole image will also be determined, and correctly reconstructed by the local algorithms, if it is known which of the singular points in the image correspond to local minima of the surface height, and if the surface heights at these points are available. If this information is not known, then there is a potential convex-concave-saddle ambiguity for the surface shape at each singular point.

We briefly describe and present experimental results for a general shape from shading algorithm which can determine this information automatically. This algorithm reconstructs shape from shading with no a priori information about the surface. It incorporates the algorithms discussed in the previous sections, but also makes use of a global consistency analysis of shape from shading similar to that of [Oliensis, 1991b]. A detailed description can be found in [Oliensis and Dupuis, 1993; Dupuis and Oliensis, 1992b]. Typical reconstruction times are less than 30 seconds on a DECstation 5000 for 128×128 images. Our experiments with the global algorithm have so far been carried out only for synthetic surfaces and images, though these are rather complex. Nevertheless, this algorithm is capable of reconstructing shape from general images with some degree of robustness, and is orders of magnitude faster than traditional variational algorithms [Horn and Brooks, 1989].

The new algorithm requires additional assumption on the imaged surface, the most important of which is that the surface height function is twice differentiable (these assumptions do not necessarily hold in our experiments). The strategy is as follows. Suppose that a singular point $s \equiv (x, y)$ is known to be an isolated local minimum of the height. The results of [Dupuis and Oliensis, 1992b; Oliensis and Dupuis, 1992] imply that the algorithms of the previous sections, if provided $z(s)$ as an initial datum, will reconstruct the correct surface in some neighborhood A_s of s . In general, from the arguments of [Oliensis, 1991b; Oliensis, 1991a], there will be other singular points s' on the boundary of the domain A_s . By continuity of the surface [Oliensis and Dupuis, 1992], the heights of these points will be correctly determined by the local algorithm.

If it is possible to identify one of these points s' as a local maxima, then we are in the same situation as before. The local algorithm can again be applied with the height provided as initial datum at the new singular point s' , which extends the surface reconstruction over the region $A_{s'} \cup A_s$. The arguments of [Oliensis, 1991b; Oliensis, 1991a] show that iterating this process will eventually yield $z(\cdot)$ at all local minima and maxima singular points, and a correct surface reconstruction over the entire image domain \mathcal{G} . For the above strategy to work, the crucial requirement is a method for identifying which of the points s' are local maxima (or minima). This is nontrivial, since the surface reconstructed by the local algorithm assuming just the one singular point s will in general reconstruct all other singular points as inflection points. Such a method is described in [Oliensis and Dupuis, 1993; Dupuis and Oliensis, 1992b].

A 128×128 synthetic surface is displayed in Figure 1. The surface was generated by creating a random surface in frequency space, masking it with a filter so as to reduce the high frequencies, and then fourier transforming to obtain the surface. The image was generated with the light from the direction $(0, .3, .95)$. Using no boundary data other than $I(\cdot)$, the surface shown in Figure 2 was reconstructed. This reconstruction took less than 30 seconds of CPU time on a DEC 5000 workstation. The surface shown is the result of reconstructing with the algorithms given in previous sections, based on the local maxima and their heights obtained with the extended algorithm.

Figure 3 illustrates the reconstruction error, the magnitude of the difference between the original surface height and that of the reconstruction. The reconstruction is good except near the edges of the image. As with our earlier algorithms [Dupuis and Oliensis, 1992b; Oliensis and Dupuis, 1992], this is due in part to the fact that the imaged surface does

not obey the boundary condition A2.1 (or its analog for reconstruction based on local maxima)—that is, the steepest descent direction for the surface at the image boundary is not everywhere into (respectively, out of) the image. This is discussed further below. The average reconstruction error in the interior of the image with $20 \leq z_2 \leq 105$ is 0.5 units, or about 2% of the height range.

We have also studied the effect of image noise. A second 128×128 surface obtained as before is displayed in Figure 4. The image was obtained using the same illumination as before, but with a random noise added at each pixel with a uniform distribution and a maximum amplitude of 0.1. Since the maximum image intensity is only $I = 1$, this is a large noise of $\pm 10\%$. The reconstruction based on the noisy image and using the recovered local minima is shown in Figure 5. Although there are large errors in some parts of the image, the reconstruction is still good over much of the image. The error in the height is displayed in Figure 6, where saturated white represents a height error of 3. The error is less than 3 units over most of the image, in comparison to the overall height range for this surface of 44 units. In the region of the image with $127 > z_{1,2} > 40$, the mean height error is just 1.6. This represents a surprising immunity to the large image noise.

In Figure 7 the error is shown for a different reconstruction from the same noisy image. It differs from the previous one in being based on the recovered local maxima. As expected, near the boundary of the image, the region of accurate reconstruction for the maxima-based method is complementary to that of the minima-based one. Since the image boundary does not respect A2.1 (for either method), the maximum-based method does better at those points near the boundary where the steepest descent direction is outward, while the minima-based one does better where this direction is inward. Together, the two methods give reconstruction with error less than three units over most of the image.

References

- [Bichsel and Pentland, 1992] M. Bichsel and A. P. Pentland. A simple algorithm for shape from shading. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 459–465, Champaign, Illinois, 1992.
- [Bruss, 1982] A. R. Bruss. The eikonal equation: Some results applicable to computer vision. *Journal of Mathematical Physics*, 23(5):890–896, 1982.
- [Dupuis and Oliensis, 1992a] P. Dupuis and J. Oliensis. Direct method for reconstructing shape from shading. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 453–458, Champaign, Illinois, 1992.
- [Dupuis and Oliensis, 1992b] P. Dupuis and J. Oliensis. An optimal control formulation and related numerical methods for a problem in shape reconstruction. Submitted to *Annals of Applied Probability*. Also UMASS CMPSCI TR 93-03, 1992.
- [Horn and Brooks, 1989] B.K.P. Horn and M. J. Brooks, editors. *Shape From Shading*. M.I.T. Press, Cambridge, Massachusetts, 1989.
- [Kushner and Dupuis, 1992] H.J. Kushner and P. Dupuis. *Numerical Methods for Stochastic Control Problems in Continuous Time*. Springer-Verlag, New York, 1992.
- [Oliensis and Dupuis, 1991] J. Oliensis and P. Dupuis. Direct method for reconstructing shape from shading. In *Proc. SPIE Conf. 1570 on Geometric Methods in Computer Vision*, pages 116–128, 1991.
- [Oliensis and Dupuis, 1992] J. Oliensis and P. Dupuis. *Direct Method for Reconstructing Shape from Shading*, pages 17–28. Jones and Bartlett, Boston, 1992.
- [Oliensis and Dupuis, 1993] J. Oliensis and P. Dupuis. A general algorithm for shape from shading. In *Proc. of the IEEE International Conference on Computer Vision*, Berlin, Germany, 1993.
- [Oliensis, 1991a] J. Oliensis. Shape from shading as a partially well-constrained problem. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 54(2):163–183, 1991.
- [Oliensis, 1991b] J. Oliensis. Uniqueness in shape from shading. *The International Journal of Computer Vision*, 6(2):75–104, 1991.
- [Rockafellar, 1970] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, 1970.
- [Rouy and Tourin, 1991a] E. Rouy and A. Tourin. A viscosity solutions approach to shape-from-shading. *SIAM J. on Numerical Analysis*, page to appear, 1991.
- [Rouy and Tourin, 1991b] E. Rouy and A. Tourin. A viscosity solutions approach to shape-from-shading. 1991.
- [Saxberg, 1989a] B. V. H. Saxberg. An application of dynamical systems theory to shape from shading. In *Proc. DARPA Image Understanding Workshop*, pages 1089–1104, Palo Alto, CA, 1989.
- [Saxberg, 1989b] B. V. H. Saxberg. A modern differential geometric approach to shape from shading. Technical Report TR 1117, MIT Artificial Intelligence Laboratory, 1989.

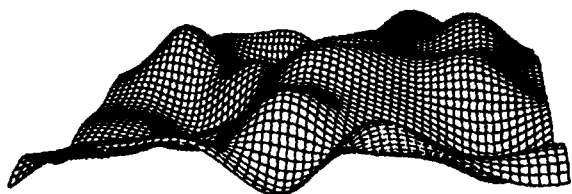


Figure 1: Complex Surface



Figure 2: Reconstruction

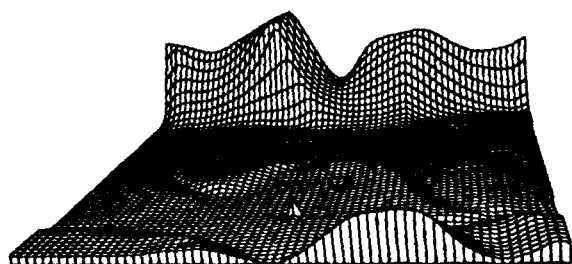


Figure 3: Difference Surface



Figure 4: Second Surface



Figure 5: Second Reconstruction

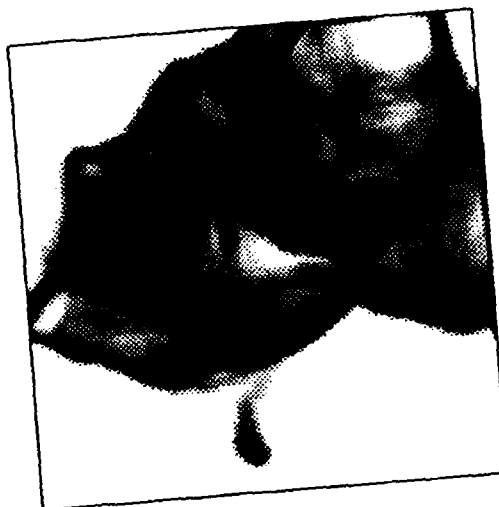


Figure 6: Reconstruction Error

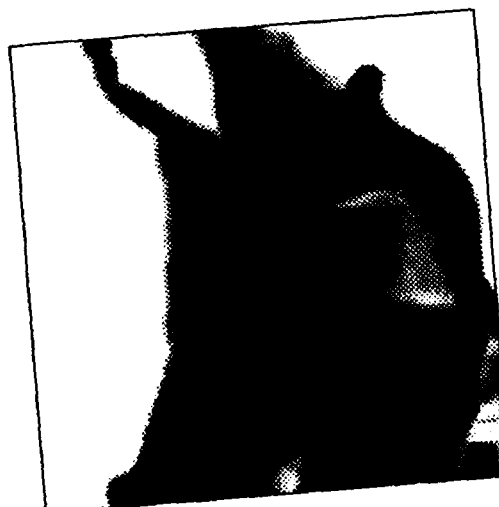


Figure 7: Reconstruction Error

Section XIX

IU Architectures and Software

Status and Current Research in the Image Understanding Architecture Program

Charles Weems, Martin Herbordt,
Rabbi Dutta, Katja Daumueller,
Glenn Weaver, Steven Dropsho

Computer Science Department
University of Massachusetts
Amherst, MA 01002

James Burrill, Richard Lerner,
Alfred Hough

Amerinex Artificial Intelligence Inc.
409 Main St.
Amherst, MA 01002

Abstract

Work on the Image Understanding Architecture (IUA)* [Weems, 1989] has advanced in three major areas in the preceding year: hardware, software, and applications. In the area of hardware, the first generation IUA (proof-of-concept prototype) has been up and running for over a year, the second generation is nearing completion, and we have started the evaluation and design process for the third generation IUA. With regard to software, we have completed the compiler for the parallel C++ class library for the low level of the IUA (along with a sequential version for workstations and a parallel implementation for the CM-2), developed most of the basic software for the intermediate level, and transported the Apply compiler for the low level to the second generation. We have transported several algorithms and applications to the first generation IUA, developed new parallel algorithms for depth from motion and extraction of straight lines, developed a parallel multi-prefix algorithm for the low-level processor, completed refinement of the DARPA IU Benchmark, and started development of the next generation of the benchmark

1. Image Understanding Architecture Hardware Development

The IUA is a heterogeneous parallel processor, consisting of three different, tightly-coupled, parallel processors. The processors are designed to address the differing requirements of the low, intermediate, and high levels of a knowledge-based computer vision system. The low-level processor is a reconfigurable mesh-connected array of bit-serial processors operating in Single Instruction Single Data (SIMD) associative and multiassociative modes (in multiassociative mode, the

processors divide into groups that can perform associative queries with independent comparands and results). Its purpose is to process image data and extract features and image data that can be represented in a symbolic form. The intermediate-level processor operates in Single Program Multiple Data (SPMD) and Multiple Instruction Multiple Data (MIMD) modes, and consists of an array of digital signal processors. The low and intermediate levels communicate through a layer of dual-ported memory. The high-level processor is a coarse-grained MIMD system that will support knowledge-based processing. Again, a layer of dual-ported memory connects the intermediate- and high-level processors. The low-level processor receives its instructions from the Array Control Unit (ACU), which also directs the intermediate level (when it is operating in SPMD mode) and coordinates interactions between the low and intermediate levels. The ACU is directed by the host and the high-level processors in a coarse-grained manner. Figure 1 shows an overview of the IUA hardware. The architecture of the first generation IUA is given a detailed treatment in [Weems, 1989].

The IUA has gone through two generations of development over the last six years. The first generation had the goal of developing a proof-of-concept prototype hardware and software system. A machine with 4K low-level processors, 64 intermediate-level processors, and a single high-level processor was constructed. The software system included both FORTH and C languages with a library of parallel functions and the ability to write expressions on parallel variables. The system has been up and running for more than a year. The prototype is controlled by a very simple ACU which was only intended to demonstrate and test the functionality of the system -- it was not intended to run full applications. However, software has since been developed to run such applications via the simple controller. The applications are coded in the C++ class library that is being developed for the low level processor of the second generation IUA. We have thus been able to develop applications for the second generation and run them on the first generation hardware.

* This work was funded in part by the Defense Advanced Research Projects Agency under contract number DAAL02-91-K-0047, monitored by the U.S. Army Research Laboratories; and by a CII grant (CDA8922572) from the National Science Foundation.

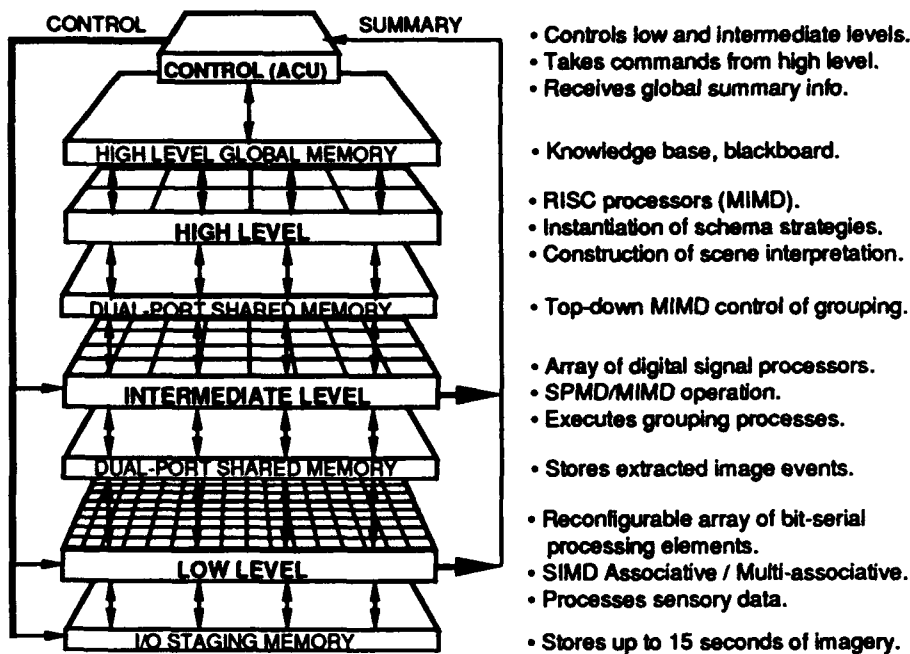


Figure 1. Overview of IUA Hardware

1.1 Second Generation IUA

With the second generation, the goal has been to develop an updated and enhanced IUA that can be embedded in the DARPA Unmanned Ground Vehicle (UGV). The basic three-level structure of the architecture has been retained, but the architecture of each level has been enhanced, the ACU has been designed to support real applications, and an I/O subsystem has been added.

The I/O subsystem has been designed to support the equivalent of 20 simultaneous sensor inputs at 512 X 512 X 8-bit resolution with automatic mapping onto the processor virtualization scheme used for the low level, with almost no latency. The I/O subsystem will also support the selection of multiple regions of interest from an image.

At the low level, the density of the processors has increased so that each chip now contains 256 elements instead of 64. The increased processor density has enabled quadrupling the number of processors while cutting the number of circuit boards in half. Memory for the low-level processors has increased by a factor of four, and I/O is greatly simplified. At the intermediate level there are still 64 processors, but they are now 32-bit, 50 MFLOPS elements with six integral 20 megabyte per second communication channels in place of the first generation's 16-bit, 5 MIPS processors which had only two 5 megabit per second channels. In the first generation, the communication channels were to be connected by a centrally controlled crossbar switch. In the second generation, each group of four

processors is directly connected to every other group. In place of a single high-level processor, a commercial multiprocessor with four or eight elements may be employed.

The second generation system is now nearing completion. The custom chips used in the low-level processor have been fabricated and tested. The 256 bit-serial processors on the chip, together with their caches, consist of roughly 600,000 transistors. The ACU for the second generation has been assembled and tested. It consists of a macrocontroller (a SPARC-based single board computer) that directs the microcontroller, which is a horizontal microengine and microcode store with queued interfaces to the processor array. The backplane, power supply and chassis for the system have been assembled and are being tested. The processor and memory boards are currently being fabricated. A design has been developed for an optional daughterboard to enhance shared-memory access in the intermediate-level processor. Hughes has indicated that the first machine should be assembled by the end of March, but without the I/O subsystem (whose construction has not yet been funded).

1.2 Third Generation IUA

Work has already begun on the analysis and design for the low-level processor of the third generation IUA. UMass has developed a system for capturing traces of programs written in the C++ class library as they execute on an abstract parallel machine. The traces are then fed to a simulation system that models hardware architectures with different features and parameters. The system, called ENPASSANT (Environment for Parallel System Simulation, Analysis and Test), allows us to

gather real performance data for different architectural configurations, and to analyze the data statistically. The performance data will then be contrasted with cost estimates for the different configurations to produce a specification for the third generation IUA. Figure 2 shows the structure of the ENPASSANT system.

Connection Machine (CM-2) implementation using C++ and PARIS. The implementation was carried out by a student who had no experience with C++, the class library, or PARIS in roughly five months of half-time effort.

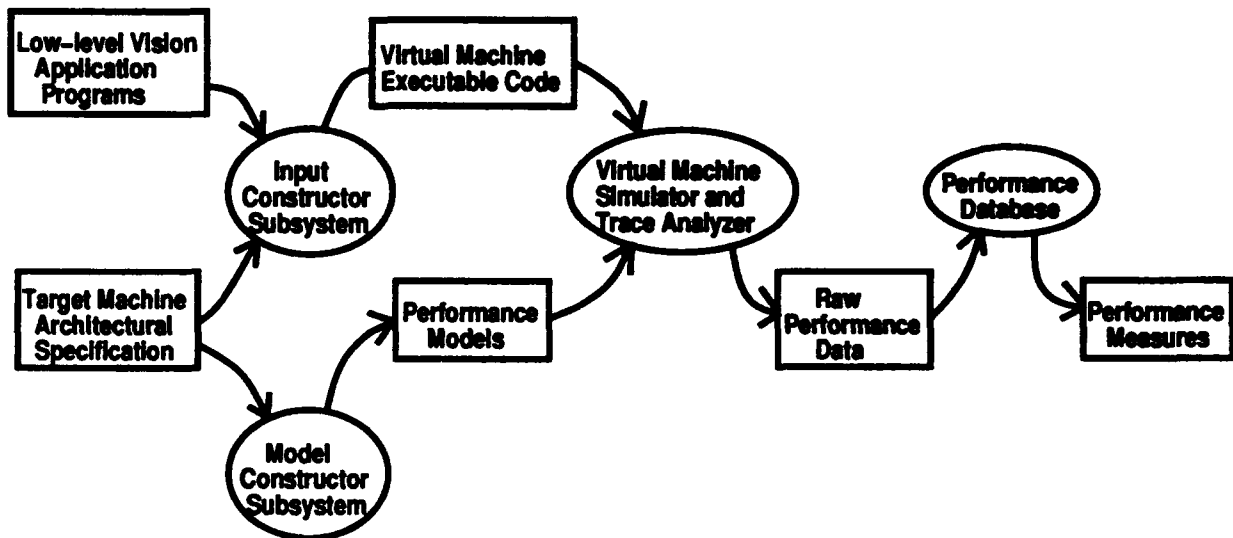


Figure 2. ENPASSANT System Structure

2. IUA Compilers and System Software

Most of the system software development for the IUA is taking place at Amerinex Artificial Intelligence Inc. (AAI), although the University of Massachusetts is developing additional software for the second generation IUA.

2.1 C++ Image Plane Class Library

AAI has completed development of the C++ class library for the low level of the IUA, including the incorporation of additional optimization code into the Gnu C++ compiler. They continue to refine the compiler optimization. The class library supports an image plane data type. Planes may be specified to be of any size, and their elements may be bit, byte, 16-bit integer, 32-bit integer, 16- and 32-bit unsigned values, and 32-bit floating point. If a plane is larger than the size of the hardware array, it is automatically mapped to a virtual processor array. In addition to arithmetic expressions on planes, neighborhood operations, and reductions, the class library also supports general routing with combining, and multiassociative processing within regions called Coteries. An automated test system has also been developed for the machine's microcode library, to facilitate regression testing of new releases of the class library run-time system.

As a test of the portability of the C++ class library to other parallel architectures, UMass developed a

2.2 Intermediate Level Software

For the intermediate-level processor, basic operating system support, multitasking, and messaging have been implemented on a TMS320C30 Single Board Computer (SBC), and recently these were transported to another SBC with two TMS320C40 processors that are configured to simulate the intermediate level of the IUA. The operating system is based on extending Spectron Microsystems SPOX™ real-time OS for the TMS320 series to support multiprocessing and interprocessor communication. Programming at the intermediate level is done with the Texas Instruments C compiler for the TMS320C40, together with library routines that support communication between processes on different processors. A multiprocessor debugger, based on GDB, has also been implemented for the intermediate level. Work is now under way to transport the KBVision™ system to the IUA, including the Intermediate-level Symbolic Representation (ISR) database.

2.3 Apply Compiler for the Low Level

UMass has implemented a version of the Apply language [Hamey, 1989], for the low-level processor of the second generation IUA. Apply is a special-purpose mini-language based on an Ada-like syntax, which facilitates the programming of local-kernel image operations. The compiler generates code compatible with the C++ class library. It permits us to easily import image processing operations written for the

CMU Warp or Intel iWarp machines. As part of testing the compiler, we have also compiled and tested the Web library of image processing routines, supplied with Apply.

3. Applications

Most of the application development has taken place at the University of Massachusetts, although Hughes has implemented some algorithms for ATR. The Hughes algorithms are proprietary and will not be described here. However, they do demonstrate some of the potential of the IUA when it is applied to real problems.

3.1 DARPA Benchmark

As recommended by the DARPA IU Benchmark Workshop participants, much of the benchmark [Weems, 1988, 1990b] has been recoded as a set of library routines which are called by the core of the benchmark. It is thus possible for implementors to make use of the image processing operations in other applications, and thereby amortize the development cost of implementing the benchmark. We have also begun developing the second level benchmark, which will incorporate tracking of moving objects over a sequence of images.

In order to reuse as much of the static benchmark as possible, the new benchmark will operate on the same type of scenes -- a 2.5 dimensional mobile of rectangles with chaff, but in the new benchmark, the mobile and chaff will be blown by an idealized wind. The goal of the new benchmark is to test system performance over a longer period of time so that, for example, caches and page tables will be filled. The benchmark will also explore I/O and real-time capabilities of the systems under test, and involve more high-level processing.

3.2 Multi-Prefix On The Low Level

The communication network in the low-level processor of the IUA is a square mesh, augmented with a second (reconfigurable) mesh, called the Coterie Network [Weems, 1990a]. The Coterie Network allows the mesh to be partitioned, for example, into areas corresponding to regions in an image. One particularly useful operation is the ability to enumerate elements within a partition or to summarize (reduce) the information in a partition to a single value. Parallel prefix is the general form of this type of operation. It is especially desirable to carry out parallel prefix in all partitions at once, i.e. to perform a multi-prefix operation.

Since feature extraction algorithms typically process thousands of regions during each of many passes over an image, the efficient computation of region parameters by a massively parallel processor requires that those regions be processed simultaneously. The difficulty when the processor is a SIMD array is in

orchestrating non-uniform data-dependent communication using a single thread of control. In this section we describe briefly how the method of "Coterie Structures" can be used to overcome this difficulty and create efficient region reduction and multi-prefix algorithms for the low level. This work is presented in detail in [Herbordt, 1992a] and [Herbordt, 1992b].

Previous reduction methods either use the standard mesh connections to route and combine packets [Herbordt, 1993], or use the coterie network to iteratively merge rectangles [Jenq 90]. The problem with the first method is that it requires a number of data transfers between neighboring processors that is proportional to the size of the largest region. For many images, this factor is the diameter of the entire array. The second method only requires a number of broadcast operations proportional to the logarithm of the size of the array. However, the proportionality constant is at least 128, and broadcast operations are somewhat more costly than neighbor transfers. Rectangle merging is thus even slower than a combining route for practical applications.

In developing efficient reduction algorithms for the CAAPP it is necessary to use the coterie network, because it reduces the complexity from being proportional to the size of the image to being proportional to the log of the size of the image. However, we use a very different approach than the typical method of iteratively merging partitions of the regions. We begin by characterizing coterie structures (configurations of contiguous PEs sharing a circuit) by their geometric properties.

Some simple coterie structures are horizontal and vertical lines, and rectangles. These structures are known to have the property of supporting reduction in $\log(N)$ broadcast operations. To these we have added the simple closed contour, boundary contour, singly vertically (or horizontally) connected contour, spanning tree, and the generic coterie structure, the coterie itself. Examples of these structures, as derived from an actual image segmentation, are shown in Figure 3. We have used a two part strategy: to create efficient reduction algorithms for whatever structures we can, and to create transformation algorithms to partition more complex structures into simpler ones. Both parts of the strategy necessarily depend on information that PEs can obtain about the network configuration in constant time, so that they can dynamically repartition the array.

Some of the basic results we have obtained in our study of coterie structures are as follows:

- Reduction can be computed on singly vertically (or horizontally) connected contour using $\log(N)$ broadcast operations, thus matching the performance of this algorithm on simple rectangles.
- Reduction can be computed on a spanning tree using $k * \log(N)$ broadcast operations, where k is a small constant.

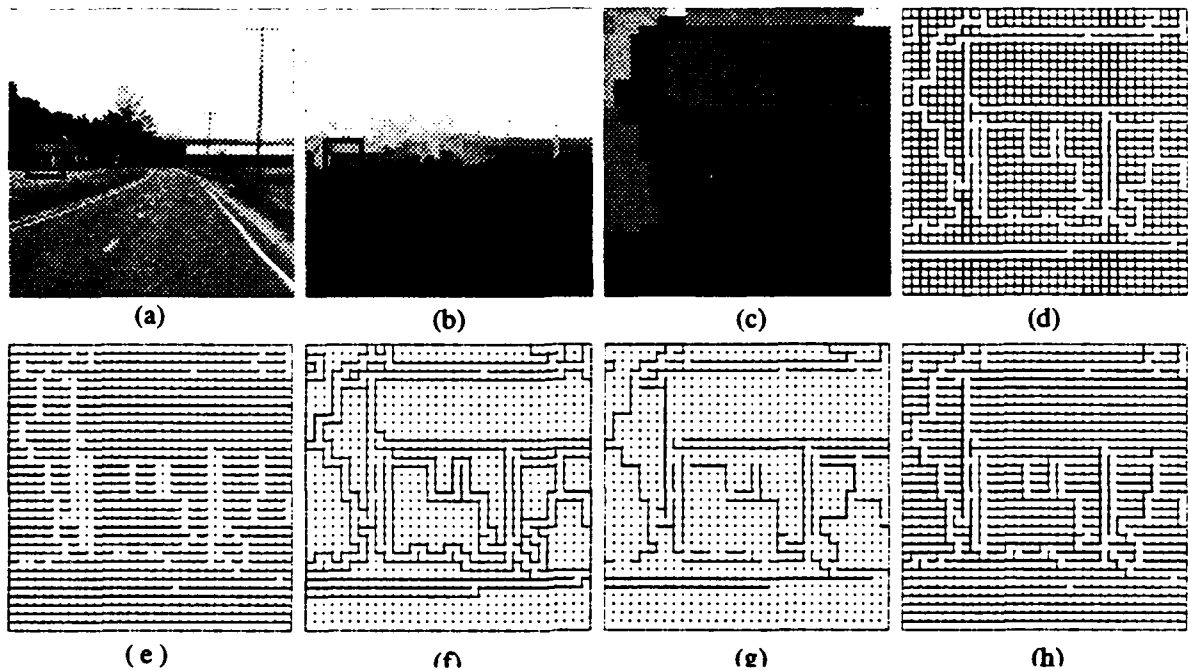


Figure 3. Example Coterie Structures. a) Image of a road scene, b) segmentation of the image, c) 32 x 32 pixel subimage of the segmentation, (d-h) represent coterie structures derived from the subimage: d) coterie corresponding to regions, e) horizontal lines, f) boundary contours, g) singly vertically connected contours, h) spanning trees.

- Any coterie can be partitioned into a minimal set of singly vertically (or horizontally) connected contours in constant time.

These results have been integrated into three efficient algorithms for the simultaneous computation of reductions on all contiguous aggregates (of arbitrary shape) in an image:

- A deterministic algorithm using $\log(N) + 4S$ broadcast operations where S is a small constant for most array partitions.
- A randomized algorithm, using an approach similar to that of Phillips [Phillips 89] and of Miller and Reif [Miller85] that uses $k * \log(N)$ broadcast operations where k is a small constant with high probability for all images. We have also shown that the algorithm is much more practical when the coterie can be preprocessed into spanning trees.
- A hybrid algorithm that combines the deterministic algorithm with associative techniques and requires only $\log(N) + T$ broadcast operations, where T has been found to be less than 20 for virtually all images.

These algorithms are significant in that they are likely to be the fastest available for reconfigurable broadcast networks for many images derived from real-world scenes.

3.3 Depth From Motion

UMass has developed a parallel algorithm for the IUA that computes a dense depth map for a scene from a pair

of images taken by a moving sensor. The only restrictions on the algorithm are that the motion of the sensor must be roughly in a forward direction, that is, the focus of expansion must be in or near the area of the images. It is also assumed that the rotational component of the motion is small (a few degrees) between the images. This would be typical of imagery obtained from a fixed camera mounted on a vehicle moving forward through an environment. The algorithm has an average error of about 8 percent in depth, as computed from randomly sampling points corresponding to objects in the scene with known distances from 21 to 76 feet from the camera. The algorithm also seems to be able to distinguish depths of objects at distances beyond what was measured in collecting the data -- trees at a computed distance of 90 feet stand out clearly from the background, for example.

The algorithm begins by determining pixel correspondences using a 3 X 3 correlation applied over the entire image for all the possible displacements between the two images, within a specified search window. It then selects the best image displacements using an interest operator, and partitions the image into tiles. Within each tile, the best of the best displacements is selected and the intersections of all of the displacement vectors are computed. A Hough transform is applied to the set of intersections and a focus of expansion (FOE) is selected from the Hough array. Once the approximate FOE is determined, an optimization process is used to estimate the best rotational parameters and then the translational motion is determined. Once the motion parameters are determined, the image displacements, together with

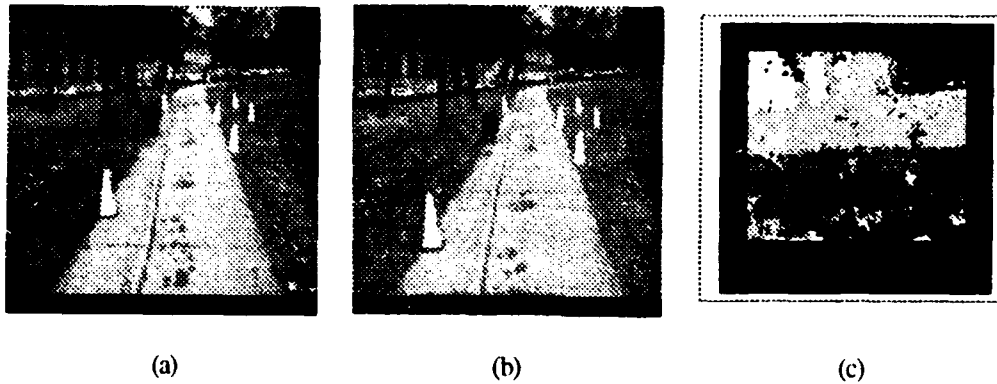


Figure 4. Result of Depth from Motion Algorithm: a) First image in sequence, b) second image, c) depth map

camera parameters, are used to determine the depths of points in the scene.

The experiments were done with fairly large displacements (four feet of forward motion between the images) so that a large (41 X 41 pixel) search window was required to establish correspondences. This results in 1681 image-to-image correlations being performed. In simulations on the second generation IUA, it was determined that the execution time will be about 0.54 seconds, of which 0.53 seconds is taken up solely by the correlations. We are thus looking into approaches in which an estimate of the motion is available or in which a series of frames with smaller displacements can be used (allowing the search window to be constrained).

Figure 4 shows two images from a motion sequence taken at Carnegie Mellon University, and a depth map that has been grey-coded into a small number of ranges. The border around the depth map corresponds to areas in the first image that have passed out of view in the second image. The other black areas represent regions in which there is a low confidence in the correspondence. For example, the area in the upper right corner is near the focus of expansion. It is possible to distinguish one of the cones and two of the distant trees in the depth

map. Figure 5 shows an enlargement of the area around the four cones in the right half of the scene in Figure 4. The grey-coding of the depth map in Figure 5 permits greater resolution of depth in that part of the map. Three of the cones produce distinct features in depth. The cone in the foreground is about 36 feet away, and the two cones in the middle are about 56 feet away. The results of this algorithm are described in detail in an article by Dutta [Dutta, 1993], elsewhere in this proceedings.

3.4 Line Extraction

UMass has also developed a parallel algorithm for extracting straight lines from an image. The algorithm begins with a Sobel operation to compute a gradient field. The gradient orientations exceeding a threshold are then divided into two sets of overlapping buckets. Using these buckets, the image is segmented into two sets of regions using a connected component labelling operation. The regions correspond to areas with a high gradient magnitude and similar orientation. Thus, they tend to be long and narrow areas surrounding candidate lines. These regions are naturally represented as Coteries in the reconfigurable mesh of the IUA.

The two sets of regions are then merged into a single

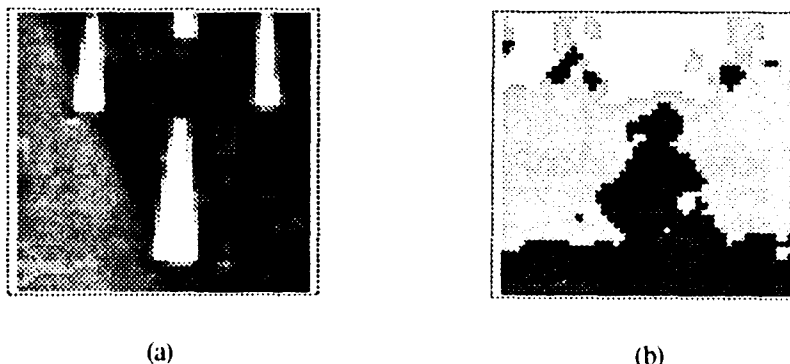


Figure 5. Enlargement of an Area in Figure 4

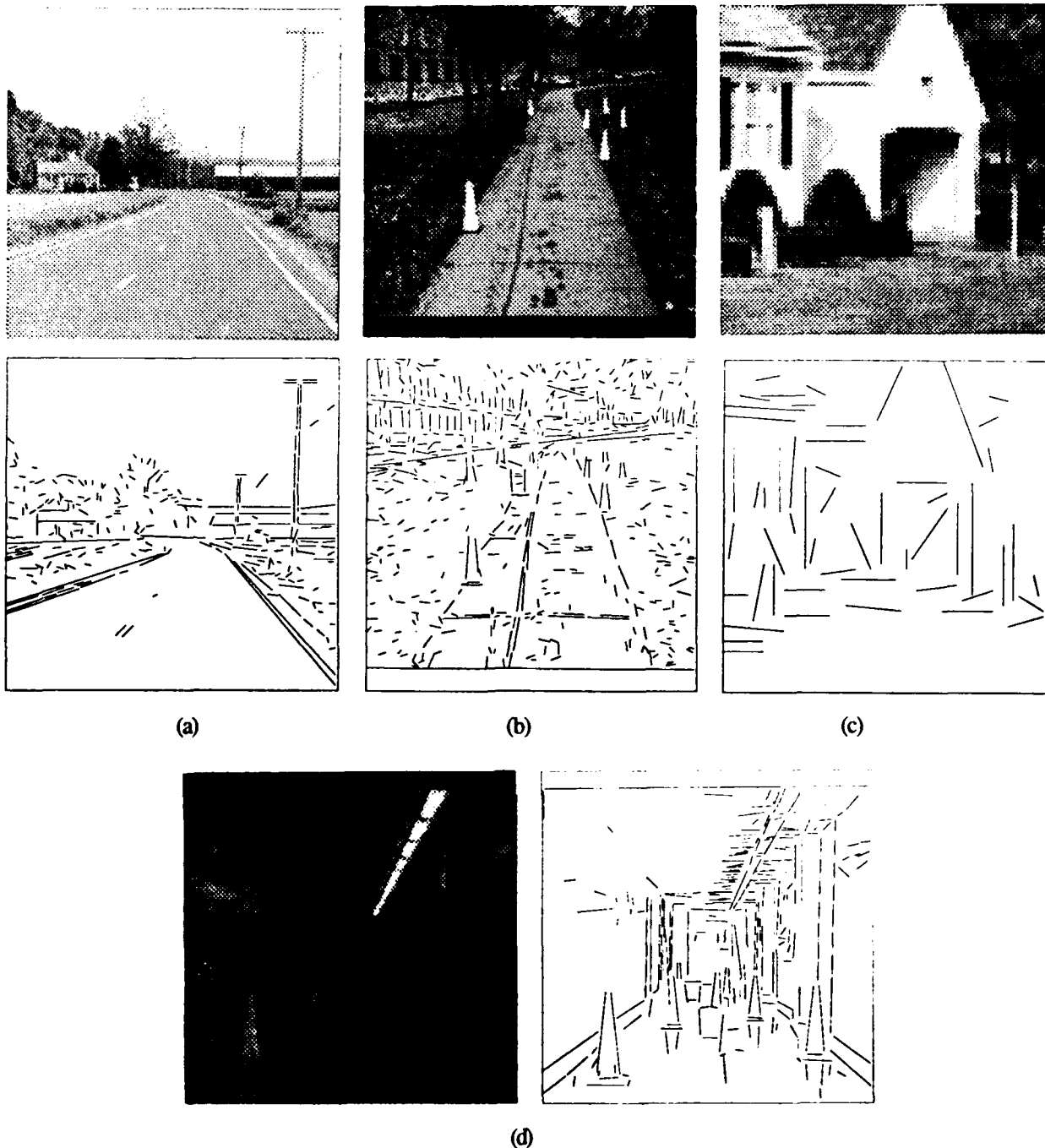


Figure 6. Results from the line algorithm executing on the IUA simulator: a) a road scene, b) the scene in Figure 4, c) a low-resolution (64 x 64) image of part of a house, d) an indoor hallway scene with obstacles

set by forming the intersection of the two sets. Pixels in these overlapping portions of regions choose to belong to the larger of the two regions, and the Coteries are appropriately reorganized. The regions are then split into three sections along their length. Within each section, the three points with the greatest gradient magnitude are selected. Then a line is fit to the nine selected points within each region. Finally, the endpoints for the lines are computed and output. Optionally, a filter based on contrast and length can be used to select different sets of lines for output. In

execution on the IUA simulator, the algorithm executes in 31 milliseconds for images that map to the array with a 1:1 virtualization ratio. The execution time will decrease once special optimizations for the 1:1 case have been added to the compiler. However, higher virtualization ratios will be approximately direct multiples of this time (i.e. a 4:1 factor results in a time of approximately 124 milliseconds). Figure 6 shows the results of applying the algorithm to six different images taken from a variety of scenes. No filtering has been applied to the lines in the figure.

4. Conclusions

During the previous year, the Image Understanding Architecture has substantially advanced toward the goal of developing a ruggedized, embeddable, reproducible hardware and software system for real-time image understanding applications. The first generation system is being used to test programs that have been developed with the second generation software environment. The second generation hardware is nearly complete, as is its basic software environment. Porting of the KBVision™ system to the second generation has begun. Research has continued in the areas of benchmarking, architectural analysis and design, and the development of both basic parallel algorithms and vision applications for the IUA.

Acknowledgements

The authors thank David Shu, Lap Chow, Dennis Finn, Howard Neely, and David Schwartz at Hughes Research Laboratories, and Charles Martin and Mitchell Smith at VillaMar Inc. for their contributions to the IUA design and development.

Bibliography

- [Dutta, 1993] Dutta, R., Weems, C.C., Riseman, E.M., Parallel Dense Depth from Motion on the Image Understanding Architecture, Proc. of the DARPA Image Understanding Workshop, April, 1993
- [Hamey, 1989] Hamey, L.G.C., Webb, J.A., and Wu, I-C, An Architecture Independent Programming Language for Low-Level Vision, Computer Vision, Graphics and Image Processing, Vol 48, 1989, pp. 246-264.
- [Herbordt, 1990] Herbordt, M.T., Weems, C.C., Message Passing Algorithms for a SIMD Torus with Coterie, Proc. ACM Intl. Symposium on Parallel Algorithms and Architectures, Crete, July, 1990.
- [Herbordt, 1992a] M.C. Herbordt, C.C. Weems, M.J. Scudder (1992): Non-Uniform Region Processing on SIMD Arrays Using the Coterie Network, Machine Vision and Applications, 5 (2), pp. 105-125.
- [Herbordt, 1992b] M.C. Herbordt, C.C. Weems (1992): Computing Reduction and Parallel Prefix Using Coterie Structures, Proceedings of the 4th Symposium on the Frontiers of Massively Parallel Computation, pp. 141-149.
- [Herbordt, 1993] M.C. Herbordt, J.C. Corbett, J. Spalding, C.C. Weems (1993): Practical Algorithms for Online Routing on SIMD Meshes, To appear in the Journal of Parallel and Distributed Computing.
- [Jenq, 1991] J.-F. Jenq, S. Sahni (1991): Reconfigurable Mesh Algorithms for the Area and Perimeter of Image Components, Proceedings of the 20th International Conference on Parallel Processing, Volume III, pp. 280-281.
- [Miller, 1985] G.L. Miller, J.H. Reif (1985): Parallel Tree Contraction and its Applications, Proceedings of the 28th IEEE Conference on the Foundations of Computer Science.
- [Phillips, 1989] C.A. Phillips (1989): Parallel Graph Contraction, Proceedings of the 1st ACM Symposium on Parallel Algorithms and Architectures, pp. 148-157.
- [Weems, 1988] Weems, C.C., Hanson, A.R., Riseman, E.M., Rosenfeld, A., An Integrated Image Understanding Benchmark: Recognition of a 2-1/2D "Mobile", Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Ann Arbor, MI, June 5-9, 1988.
- [Weems, 1989] Weems, C.C., Levitan, S.P., Hanson, A.R., Riseman, E.M., Shu, D.B., Nash, J.G., The Image Understanding Architecture, Intl. Journal of Computer Vision, 2, 251-282 (1989), Kluwer Academic Publishers, Boston, MA.
- [Weems, 1990a] Weems, C.C., Rana, D., Reconfiguration in the Low and Intermediate Levels of the Image Understanding Architecture, in Reconfigurable SIMD Parallel Processors, Hungwen Li (ed.), Prentice Hall, Englewood Cliffs, N.J., 1990. Also, COINS TR# 90-10.
- [Weems, 1990b] Weems, C.C., Riseman, E.M., Hanson, A.R., Rosenfeld, A., The DARPA Image Understanding Benchmark for Parallel Computers, Journal of Parallel and Distributed Computing, Vol. 11, No. 1 (January, 1991), pp. 1-24.

Integrating the Lisp/CLOS-C/C++ Environments

An Approach to Modular Interface Formats

Jon L White

Lucid, Inc.

tel: 415-329-8400 (ext. 5514)

jonl@lucid.com

Abstract

Our goal is to produce a major step up in the degree of integration of the C++-written and Lisp-written components of a large software project. Towards this end, we propose to make a *set of coordinated*, minor, but non-proprietary, changes to the Lucid C and C++ compilers, to the Lucid Common Lisp foreign loader, to the publicly available debugger GDB, and to extend the publicly available EMACS text editor for an increased Editor-to-Lisp communication.

We believe it is preferable to co-develop a C++ (and/or C) compiler and debugger (i.e., GDB) along with extending a Common Lisp's Foreign Language Interface, rather than trying to focus the burden of implementation entirely on one side or the other. Lucid, Inc. is in a rather unique position amongst software vendors to carry out this combined design and implementation, as it has developed and is currently marketing both a very high quality Common Lisp product and a very high quality C/C++ compiler product. In addition, Lucid is maintaining a version of GnuEmacs and is cooperating with the evolution of GDB towards this same end.

1 Overview

First, we plan to alter Lucid's C++ compiler in a way which, while not altering the semantics of the languages being compiled, will greatly increase the availability of typing and structural information both at the time of loading in the compiled object file, and at the time of execution of the program. This will provide better debugging with tools such as GDB and the Lisp "Inspector", and for increased automation in generating the interface descriptions necessary for Lisp's "Foreign Language Interface" (generally referred to as the "Foreign Function Interface", or FFI, but for years it has been more general than merely a Function-to-Function protocol.) These changes will add new information to a section of

the compiled object-file called debug-info, and will thus not impact standard linkers and loaders.

Coordinated changes in the Lucid Common Lisp foreign loader will take advantage of this information to provide automatic generation of Lisp-side class declarations. The loading of a compiled C++ class will automatically give rise to a CLOS class as a Lisp-side handle, which will be of a metaclass belonging to FOREIGN-METAClass. These CLOS classes will have an instance creation and initialization protocol on the Lisp side which invokes the corresponding *new* object allocators, and constructors, on the foreign side. Furthermore, they will support optimizable SLOT-VALUE and MAKE-INSTANCE protocols (meaning: under conditions similar to those stated for metaclass STANDARD-CLASS, the SLOT-VALUE calls can be open-coded to achieve an access timing not significantly more than a few memory cycles worse than optimal open-coded C-struct access; and similarly they will support Lisp-side optimizations such as are currently evident in the Lucid 4.0 product for the metaclass STANDARD-CLASS etc.) C++ class members returned by calls out to foreign code will be easily recognized as belonging to this metaclass, and as being members of their own direct CLOS class, upon which CLOS methods may specialize.

Additional advantages of the extra debug-info will be taken to provide consistency checking between the Lisp-side interface declarations existing at the time of object-file loading; and under an option, such declarations can even be automatically generated also, since the object files will contain a sufficiently rich amount of type-declaration information about functions and their arguments. Metaobject information will be available also to assure a Lisp-side reflection of the C++ super-classes chains, and also a dynamic INSPECT capability for C++ objects.

We will improve and automate the interface between GDB and Lisp. From the GDB side, GDB will keep and extend normal symbol-table information for each file

that is “foreign-loaded” into Lisp; and GDB will support extended commands that know about the format of Lisp data structures, so that “poking around” from GDB into a stopped Lisp image will be relatively easy to do. For example, GDB will mimic the part of the Lisp package system and PRINT and READ functions, to offer a Lisp-style interface instead of merely octal or hex “dumps”; it will be possible to set GDB breakpoints using Lisp function names, Lisp functions names will show up on the stack backtrace, and Lisp data will be printed out in a format similar to what PRINT would do. From the Lisp side, the foreign loader will prepare the necessary incremental symbol-table information for GDB, and the Lisp debugger will integrate the foreign function frames into its backtrace.

We will solidify the existing practice of Emacs-to-Lisp interfaces by extending the current style of such interfaces and embedding it into our Lisp product. A programmer desires of his development environment some aids during the step of program preparation (for Common Lisp, this usually done in using a text-editor tool). A number of different approaches towards this end have already been tried out by end-users and by other Lisp vendors; we will integrate in the better ideas, and try to work for some sort of de facto standard with the other vendors, in that the protocol on the EMACS side will be freely-available EmacsLisp code using a network-like protocol. Indeed, any program—other vendor's Lisp, or random application—can respond to such a protocol.

2 Partitioning into “Protocols”

Since this work involves the integration of familiar tools—Emacs, GDB, Common-Lisp, and C++ compilers—the question arises as to whether the communications between tools is an open protocol. We would like to make it so, and plan the following steps to assure this. In addition, we are taking this opportunity to see how much similarity we can find between the “Foreign Function Interfaces” of some of the major vendors of Common Lisp on Unix workstations.

- We plan the specification of a minimal, de facto standard for a Foreign Language Interface for Common Lisp. Primarily, this will be worked out with Harlequin, Ltd., whose basic FFI resembles Lucid's closely. We will work together to extend this as much as possible for C++ and for areas yet lacking in Lucid's published designs; in particular, the notion of a FOREIGN-METAClass for CLOS will be spelled out.
- We plan the specification of a minimal, de facto standard for a user-interface for the Emacs/Lisp communication necessary for an integrated debugging environment. Primarily, this will be worked

out with Franz, Inc., who has already placed some EmacsLisp code into the public domain for a part of an Emacs side of such an interface. The intention is that extending Emacs protocols will provide for a common interface for symbolic debugging both of Lisp and of C++.

- We plan the specification of a GDB protocol for improved interfacing and communications with a Lisp subprocess, and for making incremental additions to its symbol table. This work will generally be done in conjunction with GDB improvements being concurrently made by Lucid, Inc (for improvements to the Energize product), by Cygnus, Inc (under contract from Lucid), and by the Free Software Foundation.
- The specification of additional debug-info to be output by a C and/or C++ compiler, in support of improved typing at runtime for use such as by standard debugging tools like GDB and DBX or by a Common Lisp environment. Furthermore, techniques will be documented for specific extensions to Lucid's C and C++ compilers which will guarantee a necessary minimum of such debug-info in a compiled object-file, such that Lucid's object-file loader within the Lisp environment (the so-called “Foreign Loader”) can mechanically extract necessary interfacing information for the Lisp side of the Foreign Language Interface. These would be implemented in a product-level version of Lucid's C++ compiler, as well as in a product-level version of the Common Lisp product.

3 Coordinated Extensions of the C++ compiler and Lisp Environment

Why extend the debug-info from a C/C++ compiler, rather than tracking the compiler's decisions in a parallel, possibly Lisp-written, program? This question almost answers itself when one considers the complexity of fully correct parsing of C++ and of the current turbulence in C++ compiler technologies. It is safe to say that in general one may not inter-mix the object files from two independent C++ compilers; although one may generally do this for C compilers on any particular platform (the so-called common object file formats work because there is a common notion of how to represent nomenclatures in the object files). It is easy and straight-forward for a C++ compiler to emit a modest amount of extra information into a pre-defined debug-info format; it is easy for a Lisp's foreign loader module to continue parsing and remembering this extra information. But trying to track the compiler's decisions in a parallel parsing program will certainly be subject to the usual problems

of parallel versions—trying to keep them lock-in-step for every internal decision about runtime representations. An additional debug-info protocol emitted by the compiler is the safe step here.

Furthermore, persons working totally in the C++ world often wish that their debugger had access to more of the kinds of runtime typing information that Lisp users are accustomed to. One of Lucid's compiler wizards—himself a member of X3J16, the standardization body for C++—notes that there are continuing suggestions to that body requesting more of the runtime typing structures and capabilities in compiled C++ modules (see also Stroustrup 1992.) In short, extensions to C++ for increased availability at runtime of typing information, at least in some sort of debugging mode, will likely be of as much use and desirability to the general C++ community as to the Lisp community. While this goal is, in the long run, much greater than one project and one compiler vendor can handle, we feel that the Common-Lisp community could have much to contribute to these discussions, especially in view of a compiled Lisp/CLOS-C/C++ integration.

Finally, the use of a conventional debugger like GDB has until now been at a disadvantage because it (1) has no access to the symbols of the dynamically foreign-loaded modules, and (2) has no understanding of Lisp data formats including stack layout. We have a pilot version of a communication mechanism between Lucid Common Lisp and GDB whereby GDB's symbol table may be extended, by explicit user request, to incorporate the Lisp Foreign Loader's symbol table information also. In the pilot, the Lisp user explicitly requests the dumping of the entire "foreign" symbol into a file; and then from GDB the user explicitly overloads GDB symbol table with this file. Lucid is currently developing protocols in GDB for it to communicate with Lucid's *Energize(Tm)* product through a network socket protocol; we plan to extend these protocols so that it can automate the preparation and acquisition, incrementally, of the symbol-table information kept by Lisp foreign loader.

4 Other Research in Related Areas

While most other vendors of Lisp have some form of defined techniques for interfacing to foreign code, typically with limitations similar to that found in the Lucid product for the past five years, there have always been nagging gaps: for example, incomplete ability to interface to foreign data. Indeed, Lucid's earlier implementations, like several other vendors', only provided an interface to *functions*; by extending the work found in DEC's VAXLISP product, Lucid came up with a foreign types interface [see Sexton 1988], and automated more of the necessary data representation conversions.

About three years ago, Lucid and Sun Microsystems investigated what it would take to do what might be called a "seamless" integration of Lisp and C coding, by re-implementing a Lisp system from the beginning with the goal in mind. The effort was known by the acronym NCL, meaning "Native Common Lisp", and the outline of such a design was submitted to DARPA in response to BAA 90-15. Although the NCL work was not continued, ideas similar to it abound today in the form of one- or two-person research projects which are geared towards the demonstration of the feasibility of one or more components of it. See for example Bartlett (1988); see also Muller and Rose (1992) and Hennessey (1992).

Except for the work of Hennessey, most of these research projects have been undertaken in Scheme rather than in Common Lisp. The main attractiveness of Scheme as a vehicle for experimentation is that it can be taken to be a very small language; although most commercially available Scheme implementations have extensive development beyond the formal standard specification (e.g., macros, etc.) a very small defined set of capabilities is within the scope of a one- or two-man-year project. By contrast, a commercially rugged Common Lisp effort is well beyond that in development cost. (The successful Common Lisp vendors probably have more than 20 man years each invested in their products; and probably a guess of 100 man years might not be off the mark.) Simply because of the added complexity of real Common Lisp systems, and the variety of factors competing for trade-offs in the efficiency arena, it is difficult to say how—and even if—the limited success of these *Scheme* prototypes would generalize to an industrial-quality Common Lisp.

Although such radical re-implementation approaches are a fertile ground for research projects in advanced programming language features, we do not believe this approach to be suitable to the needs of the RADIUS projects. Our current proposal is an evolutionary approach with much less associated risk.

A similar, evolutionary approach has been started by Harlequin, Ltd., which offers a modest extension of their foreign-function interface to do some aspects of a C++ interface, including an extension to their CLOS class system to be able to reflect C++ instances into the Lisp class hierarchy. The generation of such interfaces manually (as required by the Harlequin approach) is well within currently accepted FFI technology; but still there are a good many unresolved problems with this approach, and with the Harlequin facility in particular. The most serious mismatches and unresolved problems are due to the lack of uniform runtime typing on the C++ side, and due to the need to write Lisp programs which try to second-guess what a increasing number of incompatible C++ compilers are doing about name mangling, about representation changes for opti-

mizing efficiency, etc.

Bibliography

[Bartlett, Joel] **Compacting Garbage Collection with Ambiguous Roots**, Technical Report WRL Research Report 88/2, DEC Western Research Laboratory.

[Hennessey, Wade] **WCL: Delivering Efficient Common Lisp Applications Under Unix**, Proceedings of the 1992 ACM Conference on Lisp and Functional Programming.

[Muller, Hans; and Rose, John] **Integrating the Scheme and C Languages**, Proceedings of the 1992 ACM Conference on Lisp and Functional Programming.

[Sexton, Harlan] **Foreign Functions and Common Lisp**, *Lisp Pointers* Vol. 1, No. 5, Feb 1988.

[Stroustrup, Bjarne; and Lenkov, Dmitry] **Runtime Type Identification**, Document X3J16/92-0028 (and also in the March 1992 issue of C++ Report, Vol4, No. 3, p.32)

Parallel Dense Depth from Motion on the Image Understanding Architecture¹

R. Dutta

C. C. Weems

E. M. Riseman

Computer Science Department
University of Massachusetts at Amherst
Amherst, Massachusetts 01003

Abstract

This paper describes the design and implementation of a Single Instruction Multiple Data (SIMD) depth-from-motion algorithm on the Image Understanding Architecture Simulator. Correspondences are established in parallel for two temporally separated images through correlation. The correspondences are used to determine the translational and rotational motion parameters of the camera through a parallel motion algorithm. This is done by first determining the approximate translational parameters and then constraining the search for the exact translational and rotational parameters. Finally, the dense depth map is computed from the image correspondences and the computed motion parameters. Results are analyzed for three image sequences acquired from mobile vehicles (the Autonomous Land Vehicle, the Carnegie-Mellon NAVLAB, and the UMass Denning Robot). Depths are obtained at an average accuracy of about 8% in outdoor image sequences. The depth maps are processed to locate relatively small obstacles like cans and cones to a distance of about 60 feet. Larger obstacles like hills are located even when they are much further away. Issues related to the speedup and accuracy of the computationally intensive problem of motion analysis are explored in the context of the algorithm.

1 Introduction and Motivation

Motion analysis is one of the most computationally intensive tasks in computer vision. Usually motion algorithms have relied on some form of point or feature correspondences between two or more perspective views [1-9]. These correspondence-based approaches take advantage of the image displacements induced by egomotion. Most such methods match a few hundred points or features in two tem-

porally separated images and quantitatively measure the image displacements. A consistent set of motion parameters is then determined to explain these displacements. Once the motion parameters have been determined, the depth of environmental points can be found by using their individual image displacements.

For autonomous navigation it is not enough to compute depth at a few hundred isolated points in the image. In order to detect and avoid obstacles it is necessary to find depth at a dense set of points. In addition, for practical scenarios it is desirable to process a large number of high resolution images within a small period of time. The example below calculates the number of pairs of frames that must be processed per second in a typical scenario for motion analysis (through the use of point-based or feature-based correspondence methods).

Let us assume that our camera has a resolution of 256×256 pixels and a field of view of 45° ². Let us further assume that the vehicle is moving with a speed of 50 km./hour and it is necessary to determine depth to a distance of about 10-30 metres at about 10% accuracy (except in the region immediately surrounding the focus of expansion (FOE)³ where errors in depth are necessarily high). For correspondence-based methods it can be proved theoretically [16] that in order to achieve this accuracy the vehicle must move about 2 m. forward between the processing of successive pairs of frames. Since 50 km/hr is about 14 m/sec, the motion processing system should therefore be able to process a maximum of about 7 pairs of frames

¹This work was supported in part by the Defense Advanced Research Projects Agency (via Harry Diamond Labs) by contract no. DAAL02-91-K-0047 and NSF grant CDA-8922572.

²For better recovery of rotational parameters it is best to have large field of view cameras with high image resolution. However, large field of view lenses give rise to various distortions and lower the effective resolution of the image. Our choice of camera parameters are typical of commonly available image processing systems.

³The FOE is the location in the image plane where the translational component of the image displacement is zero. If the camera moves straight ahead along its optical axis with no rotations, then the FOE is at the centre of the image.

per sec. and thus, the computation associated with each pair of frames should be approximately 140 milliseconds.

In spite of this severe requirement for speed, the problem of time complexity has not been adequately emphasized in the area of depth determination. Almost any reasonable motion algorithm needs to solve complicated non-linear equations related to the 5 independent parameters of motion. The sequential algorithms have concentrated on the use of approximations and search space reduction for the solution of these equations. Furthermore, they have not attempted to compute depth at more than a few hundred locations in the image. However, when 7 complete pairs of frames need to be processed per second and the computation associated with each frame pair involves many floating point computations a parallel design and implementation is essential. If we could completely parallelize the problem, theoretically we could achieve a speedup by a factor of 65,536 times over the equivalent sequential version with a 256×256 array of processors.

None of the earlier work in parallel motion processing [10-15] attempt a comprehensive solution to the problem of dense depth determination from a sequence of images. In the last few years a variety of processor arrays have emerged for solving low level processing in computer vision [17]. The fine grained SIMD array computers have proved particularly versatile for solving such low level tasks. The AMT DAP series, the Connection Machine and the lowest level of the IUA are three machines which embody this computational paradigm. The SIMD class of machine makes it feasible to attempt real-time solutions to the dense depth-from-motion problem. The algorithm presented in this paper is implemented on the Image Understanding Architecture (IUA) simulator. The IUA is a three layered parallel machine specifically designed for image analysis. The lowest layer of the IUA is the CAAPP, a two dimensional grid of 1-bit serial processing elements, operating in the SIMD mode.

A common scenario in ground-based navigation occurs when the vehicle moves forward by undergoing primarily translational motion along with small rotations. In this case the FOE is within the field of view. The algorithm presented in this paper is designed to take advantage of this situation. It first determines the approximate translation and then constrains the search for the exact translational and rotational parameters. *In contrast to other methods which have not demonstrated their ability to recover dense depth maps and locate obstacles, our algorithm is fast, simple and robust.*

We start by providing a brief description of the IUA which emphasizes the features most pertinent to our application.

2 The Image Understanding Architecture (IUA)

We provide a brief description of the IUA [18] with particular emphasis on the lowermost level of the machine. The IUA is made up of three levels, each having a particular type of processor:

1. *Low Level* consisting of the Content Addressable Array Parallel Processor (CAAPP).
2. *Intermediate Level* consisting of the Intermediate Communications Associative Processor (ICAP).
3. *High Level* consisting of the Symbolic Processing Array (SPA).

The CAAPP and ICAP levels are controlled by a dedicated Array Control Unit which is directed from the SPA level. The low level processors are ideal for fine-grained SIMD computing, whereas the intermediate and high level processors are ideal for Multiple Instruction Multiple Data (MIMD) computing. Our algorithm uses only the low level of the IUA because of the nature of the task. The low level or CAAPP level is a 256×256 square grid array of custom 1-bit serial processors with local memory, one-bit registers, backing store, an ALU and data routing circuitry. The bit-serial processing elements are linked through a four way (North, South, East, West) communications grid. Intra-level communication within the CAAPP can take place in several ways [18].

3 Depth from Image Displacement

This section discusses the mathematical formulation for the algorithm. Figure 1 shows a right-handed coordinate system fixed with respect to the camera. Let us also assume the right hand rule for rotations and consider the case where the camera is undergoing motion. As can be seen from Figure 1 the environmental point P , with world coordinate (X, Y, Z) , is projected onto point p , in the image plane with image coordinates (x, y) . Let f be the focal length of the camera, and denote by $\vec{T} = (T_1, T_2, T_3)$, $\vec{\Omega} = (\Omega_1, \Omega_2, \Omega_3)$ the translational and rotational rigid motion of the camera (This implies that $P' = -RP - T$ where R is the rotation matrix and P' is the new position of P after undergoing rigid motion to the next frame).

We shall use the small rotation ⁴ motion equations, and for simplicity use the following abbrevi-

⁴This means that the magnitude of rotation $|\theta| \ll 1$. Also, $\sin(\theta) \approx \theta$ and $\cos(\theta) \approx 1$ to order $O(\theta^2)$. Using the approximation $|\theta| \ll 1$ we note that even if $|\theta| = 0.1$ radians (i.e. $\approx 6^\circ$), the relative error incurred is 0.2% for $\sin(\theta)$ and 0.5% for $\cos(\theta)$. The small angle assumption is not a restrictive one in practical situations because large rotations induce such large image displacements that correspondence algorithms are unable to handle them reliably.

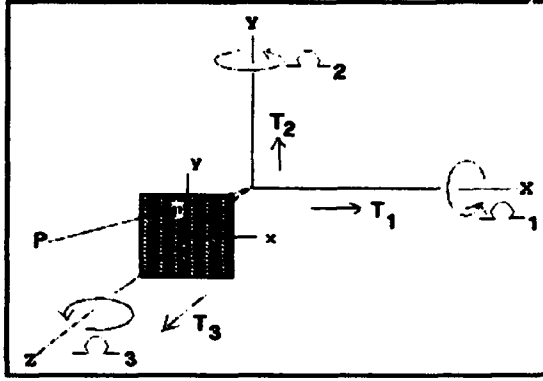


Figure 1: Coordinate System

ations:

$$\begin{aligned}
 I &= 1 + \frac{\Omega_2 x}{f} - \frac{\Omega_1 y}{f} \\
 J &= \left(\frac{\Omega_1 xy - \Omega_2 x^2}{f} \right) - \Omega_2 f + \Omega_3 y \\
 K &= \left(\frac{\Omega_1 y^2 - \Omega_2 xy}{f} \right) + \Omega_1 f - \Omega_3 x \\
 \alpha &= -fT_1 + xT_3 \\
 \beta &= -fT_2 + yT_3.
 \end{aligned}$$

With the above abbreviations the image displacement \vec{l} , induced by the motion of the camera, is given by

$$\vec{l} = u \hat{x} + v \hat{y} \equiv (u, v) \quad (1)$$

where \hat{x} and \hat{y} are unit vectors along the x -axis and y -axis respectively, and

$$u = \frac{J + (\alpha/Z)}{I - (T_3/Z)} \quad (2)$$

$$v = \frac{K + (\beta/Z)}{I - (T_3/Z)}. \quad (3)$$

The depth Z of an environmental point P can be determined from either equation (2) or equation (3). We denote by Z_x the depth determined from equation (2) and by Z_y the depth determined from equation (3). Hence,

$$Z_x = \frac{T_3 u + \alpha}{I u - J} \quad (4)$$

$$Z_y = \frac{T_3 v + \beta}{I v - K}. \quad (5)$$

Of course for perfect data, these would be equal; however, in the presence of noise, they will in general be unequal.

It is also possible to write the depth in terms of the displacement vector \vec{l} and the motion parameters by using equations (1), (2), and (3). However, the resulting expressions are cumbersome to manipulate.

In this experimental scenario the vehicle is moving forward into the scene. The motion is mostly

translational with small rotations. For this kind of "approximate translational motion" the approximate location of the focus of expansion [19, 20] can be found quite easily. This gives an initial approximate estimate for motion parameters. If the focal length of the camera is unknown, but the focus of expansion is known, then the time-to-adjacency [19] relationship is sometimes used to compute rough depth from the approximate estimate for motion parameters. The time-to-adjacency relationship gives

$$\frac{Z}{T_3} = \frac{D}{d} \quad (6)$$

where

- D = distance in pixels of point p_i , from the FOE.
- d = displacement of the point p_i .

To find better depths a more elaborate scheme is needed. From equations (4) and (5) we can see that there are two ways of determining depth for a particular set of motion parameters and image displacement. Z_x is the depth that is determined from the x -component (u) of the image displacement and Z_y is the depth that is determined from the y -component (v) of the image displacement. As u becomes close to 0, Z_x becomes hard to determine. Similarly, as v becomes close to 0, Z_y becomes hard to determine. When the movement of the vehicle is mostly forward a major portion of the image has significant magnitudes of both u and v and either Z_x or Z_y can give good depths.

For any point i in the image the depth Z_μ is computed as the average of Z_x and Z_y (except in pathological cases where either u or v is zero). It is possible to arrive at an estimate of the reliability of Z_μ by noting the difference of Z_x and Z_y from each other. The reliability ζ is defined as

$$\zeta = \frac{U(Z_x, Z_y)}{\sqrt{Z_x^2 + Z_y^2}} \quad (7)$$

where

$$\begin{aligned}
 U(x, y) &= |x + y| \quad \text{if } x \leq 0 \text{ and } y \leq 0 \\
 &= |x - y| \quad \text{otherwise}
 \end{aligned} \quad (8)$$

The reliability scale varies from 0 to $\sqrt{2}$ (no matter what the values of the two depths) with 0 being the best reliability. Qualitatively, we are testing how well the depths computed from the two components of the displacement vectors match. If both depths are positive and equal then ζ is zero (i.e. high reliability). If both depths are negative then ζ is high (low reliability). In general, when one depth is positive and the other depth is negative then reliability is poor, although not as poor as the case when both are negative.

Let ζ_i be the reliability for depth obtained at point i in the image. Let n be the total number of displacement vectors. Z_{x_i} and Z_{y_i} are the depths

computed by using the x and y -components respectively of the i^{th} displacement vector and the hypothesized values of the motion parameters. Then

$$\kappa = \frac{\sum_{i=1}^n \zeta_i}{n} \quad (9)$$

The motion parameters for which κ is minimum is the best set of motion parameters. Unlike several other approaches this optimization criterion allows us to avoid the rather difficult problem of eliminating the depths from the error functions (which has to be done somehow when the deviation between the actual and predicted image displacements must be minimized). We can call the error function κ , given by equation (9) as the *normalized absolute deviation in directional depths*. The nice property of κ is that it varies between 0 and $\sqrt{2}$. The lower the value of κ the better the minimization.

We follow this section with the algorithm for depth determination.

4 Depth Determination Algorithm

The objective is to recover reliable depths of environmental points over as large a part of the image as possible. The parallel algorithm works in the following stages:

1. Determination of *image correspondence*.
2. Selection of the best image displacements between frames.
3. Determination of the *approximate translational motion parameters*.
4. Determination of the *exact translational and rotational motion parameters*.
5. Depth determination.

4.1 Image Correspondence

The algorithm for establishing image correspondence takes as input two temporally displaced images and the maximum possible displacement at any pixel. Restricting the maximum displacement does not significantly limit the efficacy of the implementation since it is usually possible to predict it in advance. The row and column displacements at every pixel are computed through correlation matching.

The parallel algorithm for computing image displacements works as follows

Parallel Correlation

Store Frame-1 and Frame-2 of the image sequence in local processing element (PE) memory

FOR hypothesized displacements within the maximum displacement range DO
BEGIN

Shift each pizel in Frame-2 by the negative of the hypothesized displacement

Compute sum-of-absolute-differences correlation between the pizels of frame-1 and the shifted frame-2 in a spiral pattern [21].

If the correlation at a pizel is the minimum obtained until now, then store the hypothesized displacement and the current minimum correlation value in the local PE memory.

END

After all hypothesized displacements have been considered the displacement corresponding to the minimum correlation at each pixel is the image displacement at that pixel. The value of the correlation gives a measure of the reliability of the match.

It should be noted that using sum-of-absolute differences rather than the sum-of-squared differences as the correlation measure saves time by avoiding multiplications. The repeated correlation computations are the most computationally intensive part of the algorithm and this is an important efficiency consideration. Integer representations are also preferable because floating point computations are costly in a bit-serial processor.

4.2 Selection of best image displacements

The coterie network of the CAAPP is used to partition the image into equal divisions [18]. For example, in one experiment the 256×256 image was divided into 64 sub-regions of 32×32 pixels each. In each sub-region the pixel which has the most reliable displacement vector is selected in parallel. Hence there are as many selected pixels as there are sub-regions. The displacement vectors at these selected pixels are called "selected" displacement vectors. The FOE and motion parameters are determined with the selected displacement vectors.

4.3 Approximate Translation

In dynamic imaging situations where the sensor is undergoing primarily translational motion with a relatively small rotational component, *approximate translational motion algorithms* may be effective in determining approximate depth [20]. By restricting the processing to the two dimensions of translational motion, there is a tremendous reduction in complexity from the five dimensions (excluding the scaling component of sensor velocity) of general motion.

The FOE recovery algorithm works as follows. First it draws a line through each selected displacement vector in the image. Let these lines be called "extended" displacement vectors. Then it finds all

the possible intersections of the extended displacement vectors and votes in a Hough transform array corresponding to each intersection [22]. The parameter space for the Hough transform is given by the image coordinates where the extended displacement vectors can intersect each other. The number of votes for each intersection is an increasing function of the length and confidences of the displacement vectors forming the intersection. This ensures that longer displacement vectors are more heavily weighted and that more reliable vectors are also weighted more heavily. The point where the maximum number of intersections occur is the approximate FOE. A contiguous region which surrounds the approximate FOE and includes at least ρ fraction of the votes is the region where the exact FOE is likely to be present.

The parameter space for the Hough transform is spread over all the processing elements of the CAAPP. The intersections have an x -coordinate and a y -coordinate. Since the CAAPP is a two dimensional array, it is easy to map each possible intersection into a distinct PE. The local memory of each PE contains the number of votes for an approximate FOE. The contiguous region where the exact FOE might be located is determined by summing up in parallel the votes gathered by neighbors [22].

4.4 Exact Translation and Rotation

Once the region in which the exact FOE can lie is determined the exact translational and rotational parameters can be computed by the optimization method stated in equation (9).

For example let the approximate FOE be at (70,55) with the exact FOE lying within 10 pixels of the approximate FOE. Then a square whose sides are 20 pixels is formed with (70,55) as the intersection point of its two diagonals. Then FOE's are hypothesized at each pixel bounded by the square (i.e. starting at (60,45) and ending at (80,65)). At each hypothesized FOE the normalized absolute deviation in directional depths κ , is computed for the three dimensions of rotations. The rotations corresponding to the minimum κ are the optimal rotational parameters corresponding to the hypothesized FOE. The minimum κ is determined among all the hypothesized FOE's. The translation and rotation corresponding to minimum κ are the exact motion parameters.

4.5 Depth Determination

Once the motion parameters are obtained the depth at each point in the image can be found by using the image displacements and the intrinsic camera parameters. Since each pixel of the image is represented by one processing element in the CAAPP, this stage is totally parallelized. The equations used for this purpose have been described in Section 3.

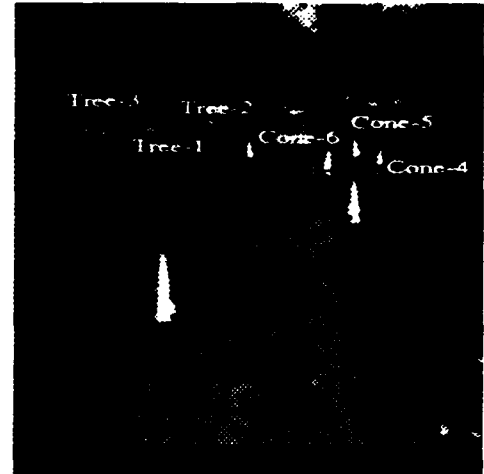


Figure 2: First frame of the Sequence.

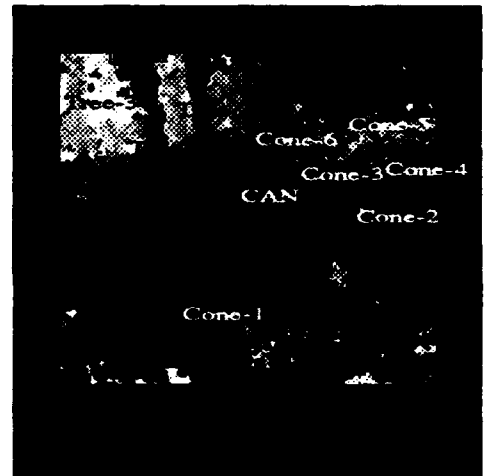


Figure 3: Computed depth map.

Object	True depth (feet)	Computed Depth sample		
		#1	#2	#3
cone1	21	22	23	25
cone2	36	35	44	33
cone3	56	45	53	54
cone4	56	54	55	56
can	46	41	44	43
cone5(*)	76	96	66	70
cone6	76	62	67	55
Tree1	-	50	51	51
Tree2	-	58	59	57
Tree3	-	90	91	91

Table 1: Depth Values of some environmental points.(*) Cone5 is near the FOE and its depths are erroneous.

5 Experiments

The algorithm for parallel depth determination was coded on the IUA simulator. This section contains the results obtained on several image sequences.

Carnegie Mellon NAVLAB sequence - The first set of experiments used a sequence of twenty images. The images were collected on the Carnegie Mellon NAVLAB. The first image of the sequence is shown in Figure 2. The vehicle was made to move in an approximately straight line such that the distance between frames was 2 feet. The field of view of the camera was 45° and 256×256 images were collected. In order to determine the ground truth for environmental objects, traffic cones and a can were placed at measured distances (ranging from 21 to 76 feet). Obviously with a total movement of the vehicle of 40 feet, some of the cones disappeared from the scene in later frames. Figure 2 shows environmental objects whose depths had been hand measured (black dots), along with the rest of the scene with objects whose distances were not measured (e.g. trees). It should be noted that in general the scene is quite complex because of the presence of large homogeneously textured regions like road and grass, and the occlusion of the distant buildings through trees.

The quantitative results for the known environmental objects and some unknown objects are shown in Table 1. Three visually selected pixels were marked on each object. The value of depth returned by the algorithm at each of these pixels are recorded in Table 1. These recordings are referred to as "samples" in the table. From the table the average error in depth for the known objects is computed to be about 8%⁵. This corresponds quite well to the theoretical limits on depth determination presented in our previous work [16].

The depth map obtained by using the algorithm between the first and third frame of the sequence is shown in Figure 3⁶. Complete separation of obstacles (e.g. cones, can) is not possible from the depth map because part of the surroundings of an obstacle in the depth map is at almost the same depth as the obstacle. This is very different from an intensity image where all regions of an obstacle usually have a different image intensity from its surrounding. The darker the color the closer the environmental point is to the camera (*Since the gray-level representation of depth on a printed page is poor, the results will be presented in various representations. On a high resolution display device the depth maps appear a lot better to the human eye.*). It should be noted that white on the depth

map indicates points at which depths are over 120 feet. Black indicates points where depth cannot be computed reliably (e.g. periphery of the image that is not visible in both images, points where displacement vectors are small or erroneous thereby giving rise to wrong depths etc.). Parts of cones, can and trees stand out in the depth map in subsequent representations.

Figure 4(a)-(l) shows the two frames, some intermediate results and several depth maps at different regions of the image. An explanation is necessary for the legends which illustrate the depth maps. The legends are histograms of depth maps. The x-axis shows the depth values and the y-axis indicates the number of pixels with a particular depth value. The shading of the histogram corresponds to the shading of the depth map. For example, in figure 4(e) (whose histogram is shown in figure 4(f)), the lightest region show depths of over 80 feet. The top of cones and can stand out in the depth maps. Observers unaccustomed to seeing depth images often attempt to compare them with intensity images and draw erroneous conclusions. In the case of a cone standing upright on the ground in 3-D, the 2-D image will have the following characteristics:

- The top of the cone is surrounded in the image by locations which are much further away than the cone.
- The bottom of the cone has almost the same depth as the ground in front of it and to its sides.

Hence, in the depth map the bottom of the cones and can merge with the surrounding ground whereas the top clearly stands out from its surroundings. Virtually all the major obstacles at least partially stand out in a magnified depth map. We have not magnified all the obstacles in Figure 4 because of space limitations. Nevertheless, it is quite clear from the figures that portions of the two trees, the can and the cones clearly stand out in the depth map. Thus in addition to the quantitative depths the obstacles are also detected. It can be seen that even small obstacles like cones and can are detected quite robustly by this algorithm to a distance of about 60 feet. Larger structure like trees can obviously be detected more easily even when they are quite far away.

Autonomous Land Vehicle sequence- A second experiment was done on a sequence collected via the Autonomous Land Vehicle. The data collection process is detailed in [23]. Preliminary results on this sequence are shown in Figure 5. It can be seen that the mountain which is rather far away is clearly identified.

Umass Denning Robot sequence- A third experiment was done on a sequence collected via the Denning robot at the University of Massachusetts at Amherst. This image was taken indoor under poor lighting conditions. The robot moved 1.95 feet

⁵The mean depth of each known object is the average of the three samples that have been shown in Table 1

⁶The correspondences in the experimental section are from a hierarchical correlation based algorithm which has not been completely parallelized on the IUA. The completely parallel IUA implementation gives somewhat inferior results for depth.

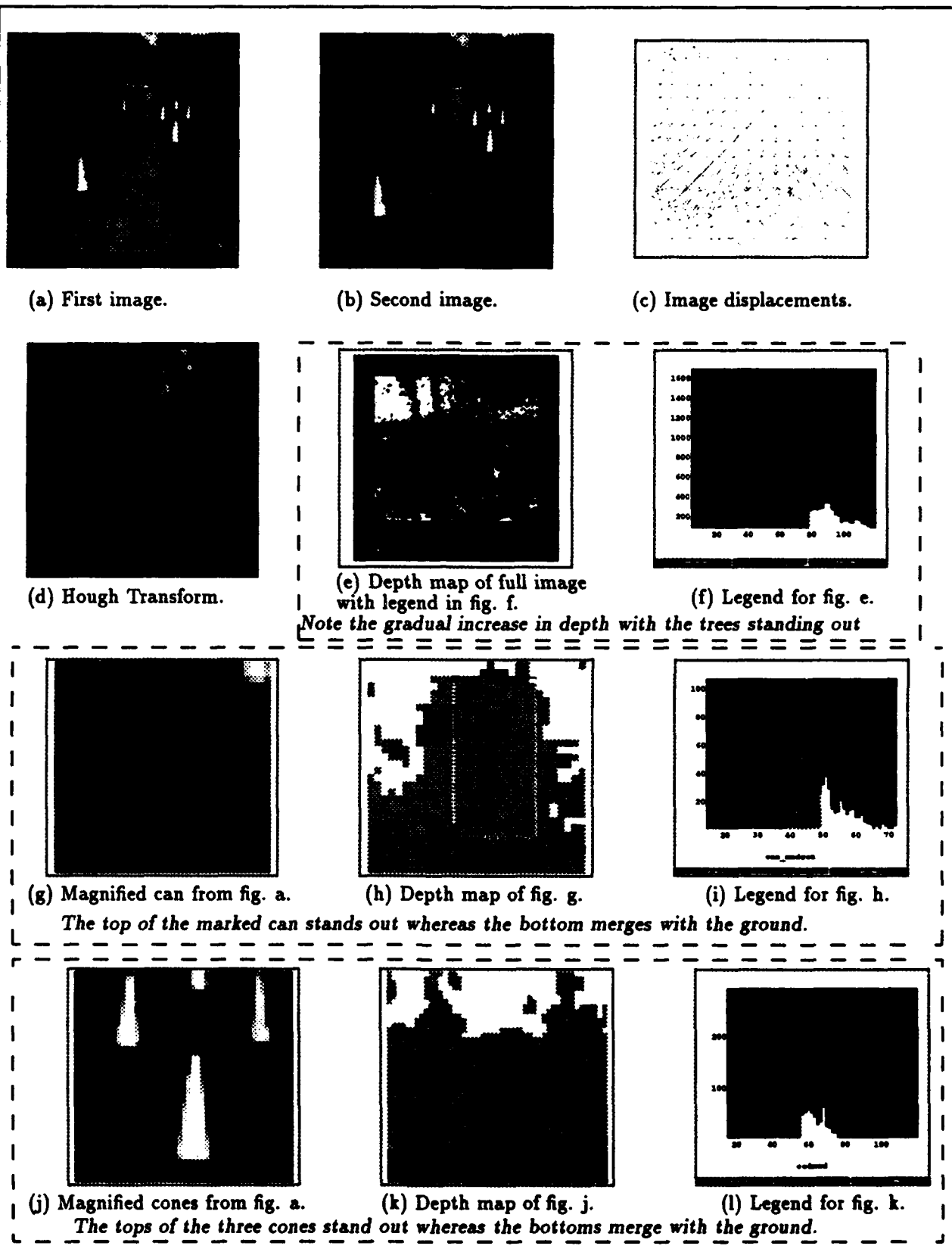


Figure 4: Results for the CMU sequence.

between frames. The results for the depth maps are shown in Figure 5.

Timings

Before presenting the timings let us caution the reader that these are results obtained by running our algorithm on the simulator of an experimental parallel machine under development⁷. For the Carnegie Mellon NAVLAB sequence the PE cycles taken for the various stages are as follows:

	CYCLES
Correspondence stage	5315983
Trans. stage after vector selection	78513
Trans. and Rot. stage with depth	35318
Sum of the above three stages	5429814

This gives us an estimate of about 0.54 sec. for running the algorithm on an IUA running at 10 MHz. The majority of the time in our algorithm is consumed by the computation of correspondences in searching over a 41×41 window for possible displacements. Reducing the size of the search window, for example by assuming constraints on the translational magnitude and direction, along with some rudimentary knowledge of depth, can make the algorithm much faster.

It should be noted that with sophisticated mechanical devices like land navigation systems, gyroscopes, inertial navigation systems etc. it is often possible to constrain the estimates of the motion parameters. If such knowledge is available then this algorithm can be speeded up considerably because the range for hypothesized FOE's and the range of rotational parameters become smaller.

6 Conclusions

This research demonstrates the feasibility of fast depth determination through the use of motion algorithms under the SIMD mode of processing. The algorithm is relatively simple because it has been designed specifically for the case of a vehicle moving mostly ahead with small rotations. It is a common scenario in navigation. Even though it is algorithmically simple, the depth computations are robust. Furthermore, there is usually no manual selection among multiple minima for computing motion parameters. Some widely used algorithms (like that proposed by Horn [4]) frequently require manual selection of the optimal motion parameters because of the presence of multiple minima. In this algorithm only the region around the approximately determined FOE is searched for computing optimal

motion parameters. This does not normally give rise to multiple minima for small rotations.

The system can compute approximate depth even when the focal length of the camera is not known. This is done by using the time-to-adjacency relationship after determining the approximate FOE. The drawback of the system is that the large number of floating point operations are expensive for a SIMD machine. However, the large number of processors more than compensate for this. The algorithm is being refined for getting faster and better correspondences. Methods for taking care of larger rotations are also being investigated.

To summarize, the key contribution of this system is that it is able to recover dense depth maps and locate obstacles quickly, simply and robustly. With reasonable assumptions, the algorithm can run in real time on the IUA.

References

- [1] H. C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. In *Image Understanding 1984*, pages 179-193. Ablex Publishing Corporation, 1984.
- [2] R. Tsai and T. S. Huang. Uniqueness and estimation of 3-d motion parameters of rigid bodies with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 13-27, January 1984.
- [3] O. Faugeras, F. Lustman, and G. Toscani. Motion and structure from motion from point and line matches. *International Conference on Computer Vision*, pages 25-34, 1987.
- [4] B. K. P. Horn. Relative orientation. *International Journal of Computer Vision*, 4(1):59-78, 1990.
- [5] J. W. Roach and J. K. Aggarwal. Determining the movement of objects from a sequence of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 554-562, November 1980.
- [6] Gilad Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 384-401, July 1985.
- [7] A. Bruss and B. K. P. Horn. Passive navigation. *Computer Vision Graphics and Image Processing*, pages 3-20, January 1983.
- [8] H. C. Longuet Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, pages 133-135, September 1981.
- [9] J. Weng, T. S. Huang, and N. Ahuja. Motion and structure from two perspective views: Algorithms, error analysis, and error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 451-476, May 1989.
- [10] C. Koch, H. T. Wang, B. Mathur, A. Hsu, and H. Suarez. Computing optical flow in resistive networks and in the primate visual system. *Proceedings IEEE Workshop on Visual Motion*, Irvine, Calif., pages 62-72, March 1989.

⁷The timings given are estimates and are subject to change. The estimates include only the time taken by the CAAPP. The time required for front-end processing is not available. Since the rotation computation part is still in the process of development, there are some sequential aspects involving front-end processing. This will be reduced in subsequent implementations.

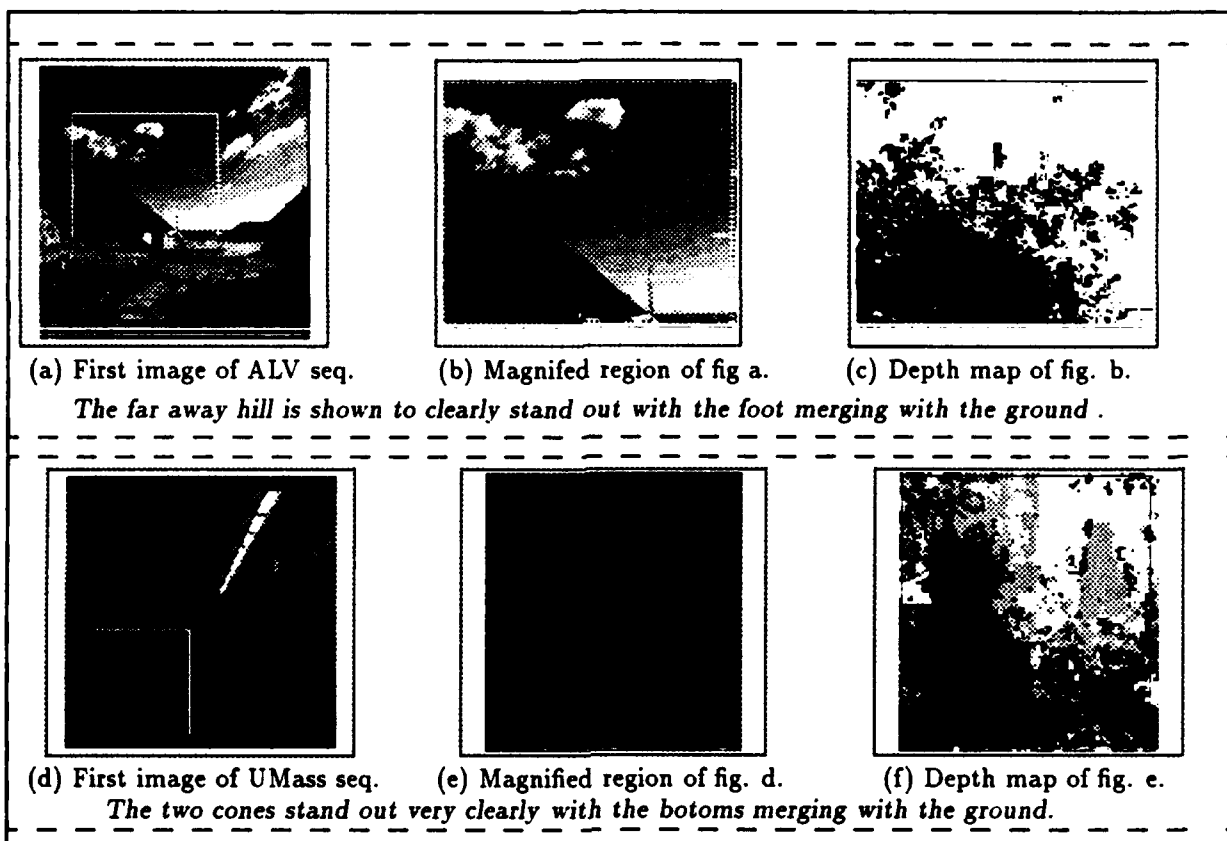


Figure 5: Results for the ALV and the UMass Hallway sequence

- [11] S. Gong and M. Brady. Parallel computation of optic flow. *European Conference on Computer Vision*, pages 124-133, 1990.
- [12] F. Heitz, P. Perez, and P. Bouthamy. Parallel visual motion analysis using multiscale markov random fields. *IEEE Workshop on Visual Motion*, October 1991.
- [13] A.N.Choudhary, M. K. Leung, T.S. Huang, and J.H. Patel. Parallel implementation and evaluation of motion estimation system algorithms on a distributed memory multiprocessor using knowledge based mappings. *International Conference on Pattern Recognition*, 1990.
- [14] H. H. Bulthoff, J. J. Little, and T. Poggio. A parallel motion algorithm consistent with psychophysics and physiology. *Workshop on Visual Motion*, 1989.
- [15] D. T. Lawton M. Streenstrup and C. Weems. Determination of the rotational and translational components of a flow field using a content addressable parallel processor. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 401-404, 1983.
- [16] R. Dutta and M. A. Snyder. Robustness of correspondence-based structure from motion. *International Conference on Computer Vision*, December 1990.
- [17] P. M. Dew, R. A. Earnshaw, and T. R. Heywood. *Parallel Processing for Computer Vision and Display*. Addison-Wesley, 1989.
- [18] C. Weems, S. Levitan, A. Hanson, E. Riseman, J. Nash, and D. Shu. The image understanding architecture. *International Journal of Computer Vision*, Volume 2(3):251-282, 1989.
- [19] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall Inc., 1982.
- [20] R. Dutta, R. Manmatha, E. Riseman, and M. Snyder. Issues in extracting motion parameters and depth from approximate translational motion. *Proceedings DARPA Image Understanding Workshop*, Volume 2:pages 945-960, April 1988.
- [21] Charles C. Weems. An algorithm for a simple image convolution on the titanic content addressable parallel array processor. Technical report, COINS Dept., Univ. of Massachusetts at Amherst, 1983. COINS Technical report 83-07.
- [22] R. Dutta. *Depth from Motion and Stereo: Parallel and Sequential Algorithms, Robustness and Lower bounds*. PhD thesis, Computer Science Dept., University of Massachusetts at Amherst, 1993. (Forthcoming).
- [23] R. Dutta, R. Manmatha, L. R. Williams, and E. M. Riseman. A data set for quantitative motion analysis. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 159-164, June 1989.

ISR3: A Token Database for Integration of Visual Modules

Bruce Draper, Allen R. Hanson, and Edward M. Riseman
Computer Vision Research Laboratory
Dept. of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Abstract¹

Experimental systems which must function in real-time application domains place significant constraints on the underlying infrastructure and support tools and on the design of these tools. One of the more important tradeoffs involves efficiency versus flexibility. This paper discusses the design philosophy of MoPLSE - the Mobile Perception Laboratory Software Environment designed to support real-time autonomous navigation.

1. Introduction

The convergence of computer vision and robotics has created the need for a new kind of software environment which supports real-time computing. The need is becoming particularly acute as applications in autonomous vehicle and flexible manufacturing become more sophisticated. Although development environments for computer vision, such as KBVision™, ImageCalc and Khoros (and soon the DARPA IUE) do exist, these environments are not well-suited to real-time applications. Conversely, robotic environments, such as VxWorks, do not supply much support for computer vision.

This paper describes the preliminary design of a portion of MoPLSE, the software environment supporting computer vision research on board the University of Massachusetts' Mobile Perception Laboratory (MPL; see below). More important than MoPLSE itself, however, are the design decisions that underlie it. Designing a software environment for real-time vision involves cost/benefit tradeoffs. On the one hand, because the environment is for research, it should be general enough to support not only currently available vision algorithms, but unanticipated future advances as well. On the other hand, one cannot afford a profligate generality: mobile

robots and autonomous vehicles must act and react in real time, so their software environments must be highly efficient.

The challenge of MoPLSE, therefore, was to build an environment that satisfies the needs of the growing real-time vision community without getting "bogged down" with rarely-used features or excessive overhead. This required making hard choices about what features are necessary for real-time vision. To some extent, these decisions depended on our application and on our general approach to computer vision. We therefore describe MoPLSE within the context of MPL and its algorithms. In a larger sense, however, the intuitions behind MoPLSE are based on fifteen years of experience in building experimental vision systems at the Univ. of Massachusetts [Draper 89], and in building tools to support that research [Kohler 82, Brolio 89]. MoPLSE differs from previous vision environments because the constraints of real-time vision forced us to prioritize needs more strongly than in previous systems, and although the details of MoPLSE will probably change over time, the need to prioritize will remain.

2. The Mobile Perception Laboratory

The Mobile Perception Laboratory (MPL) is a modified U.S. Army HMMWV with computer-controlled steering, acceleration and braking. MPL was built as part of DARPA's Unmanned Ground Vehicle (UGV) project to test algorithms for directing an autonomous military vehicle under real-world conditions. Because of the generality of the autonomous operation task, the algorithms being tested on the MPL include many of the basic visual tasks of interest to the image understanding community. In particular, the MPL was designed to support UMass research in *landmark-based navigation* [Fennema 91, Kumar 92, Thomas 93a,b, Dutta 93], *obstacle detection* and *automatic model acquisition* [Sawhney 92a,b, 93], and massively parallel computation on the Image Understand Architecture [Weems 89]. The landmark navigation task requires that the MPL recognize landmarks (stationary recognizable

¹This research has been supported in part by the Defense Advanced Research Projects Agency under TACOM contract number DAAE07-91-C-R035, by the National Science Foundation under grant CDA-8922572, and by RADC under contract number F30602-91-C-0037.

features and objects) from a map and determine the position and orientation of the vehicle relative to the landmark (i.e. pose determination).

The obstacle detection task, whether on-road or off-road, requires that MPL model the surfaces in front of the vehicle and determine whether that surface is flat enough to traverse. The model acquisition task has two sub tasks. The first is to automatically acquire new landmarks from an initial (and possibly sparse) set of landmarks. The second is to improve the vehicle's map of an area as it drives through (e.g. for use by subsequent vehicles) or, in the extreme case, to build up a map of an area from scratch. In addition, MPL also serves as a secondary test vehicle for research being conducted at other sites on road following, off-road navigation and stereo.

Physically, MPL is a modification of the ambulance (996) model of the HMMWV (see Figure 1). In front it has a pair of forward-looking, black-and-white CCD cameras for road following and obstacle detection. On top it has a pan-and-tilt controllable stabilized platform, with a 360 degree field of view, that carries two color CCD cameras and an infra-red sensor. One of the color cameras has a wide-angle lens for rapidly finding landmarks, while the other has a telephoto lens for focusing on a landmark and determining its pose. In the back, MPL contains a complete computer laboratory, including power, air conditioning and facilities for two researchers (in addition to the safety driver). On-board computing resources include a Motorola 68030 processor for directly controlling speed and direction, a Datacube MaxVideo 20 image processor, and a four-processor 340GX workstation from Silicon Graphics. Space and power is also provided for the Image Understanding Architecture, a massively-parallel heterogeneous processor being developed jointly by the University of Massachusetts and Hughes Research Laboratory [Weems 89].

3. Software Environment Specifications

The first step in designing any software environment is to determine what features are needed. In the case of the MPL, we needed to integrate many different styles of visual algorithms within a single real-time environment. The ALVINN road-following program [Pomerleau 90,92], for example, is a neural-net algorithm developed at CMU that takes a reduced (30x32)

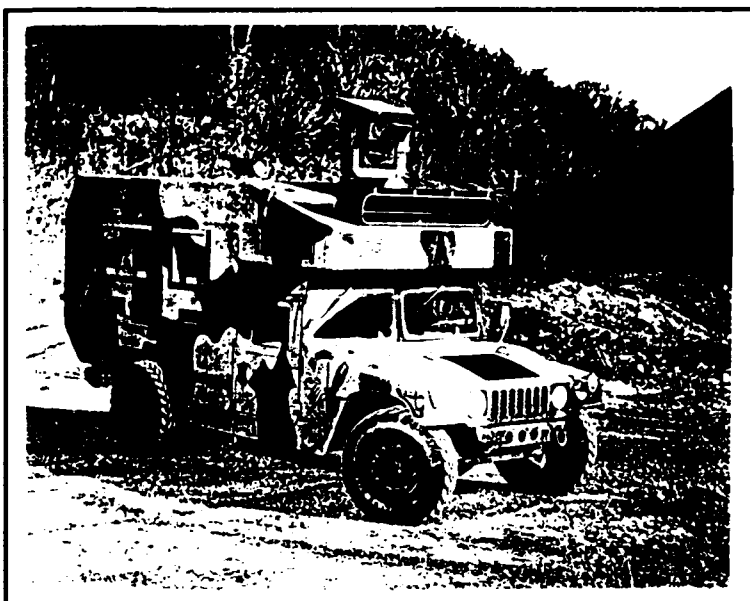


Figure 1. The UMass Mobile Perception laboratory

image as input and returns a steering angle. The centerpiece of the landmark recognition system, on the other hand, is a geometric model matcher that matches 2D line segments at the highest resolution available, extracted from an image, to 3D model edges and determines the pose of the camera relative to the landmark. It was obvious, therefore, that we needed to support a wide range of algorithms and data structures.

The decision was made, therefore, to create an environment based on the idea of visual modules. ALVINN, for example, would be one module, while the geometric model matcher [Beveridge 92] would be another. Additional modules were quickly defined for camera control, line extraction [Burns 86], pixel-based tracking, symbolic line tracking [Williams 88, Sawhney 92a], and many other visual tasks. Each module is defined in terms of its input and output, and a set of tuning parameters that allow the on-board researchers to modify the performance of a module. The primary function of the on-board software environment, then, is to facilitate the assembly of existing modules into aggregates which perform high level functions and to "optimize" the performance of this new function by tuning the parameters of the component processes.

Part of the reason for this decision was the realization that the on-board environment did not have to support the development of new modules. In general, basic algorithm development occurs in the indoor laboratory using data collected from previous experiments. The long hours of algorithm

design, implementation and initial debugging do not occur on the vehicle. Only when modules have been implemented and initially debugged are they put on the MPL, where they can be tested, revised and have any newly discovered bugs removed. As a result, the module development environment is different from the on-board real-time environment, and is not constrained by its real-time demands.

Within the laboratory, therefore, vision researchers are free to use any of the existing computer vision environments. At the University of Massachusetts the most common choice for module development is KBVision, although the Khoros environment is also available. As discussed below, tools have been developed to help researchers convert their algorithms from KBVision or Khoros to run in the MoPLSE environment.

The design of MoPLSE therefore focused on how to support the controlled invocation of visual modules, and how to pass data from one module to another. Before discussing these decisions, however, it is helpful to review the KBVision and Khoros development environments to see how any of MoPLSE's features are actually real-time adaptations of features already available in these products.

3.1. KBVision

KBVision [Williams 90] is a computer vision software development environment developed by Amerinex Artificial Intelligence (AAI). Although KBVision supports the development of visual modules in several ways, its most unique and salient features are the ISR visual database and the Image Examiner graphics tool.

The ISR is a symbolic database for visual data originally developed at the Univ. of Massachusetts [Brolio89]. ISR was motivated by the belief that computer vision requires more than image-like arrays of numerical data; it depends on symbolic representations of abstract image events such as regions, lines, and surfaces² and on mechanisms for efficiently accessing the objects under various types of constraints (such as spatial proximity). Visual modules operate on these symbolic records, called *tokens*, or on sets of tokens, and generally produce either new tokens, fill in fields of existing ones, or both.

The ISR serves as the data repository at the center of the system. Visual modules access data, in the

form of tokens, in ISR and create new tokens which they add to the database. (Strictly speaking, it is a misnomer to call ISR a database; although it does have facilities for writing data sets to files and reading them back in again, file access is too slow a data transfer mechanism for real-time vision. The ISR keeps its data in memory, and should therefore be called a data store.) What separates ISR from other systems is that its data access routines have been optimized for computer vision.

In particular, ISR must provide functions for accessing tokens by spatial location and by feature value, as well as by name. Such access functions are important because tokens are rarely isolated entities: typically, they have important symbolic, numeric and logical relations to each other. Hierarchies of tokens are important, as when surfaces are defined by ordered sequences of lines, and lines are defined by pairs of endpoints. It is therefore important to be able to access all the points in a given model, for example (a form of indirect named access). Just as frequently, tokens are organized less rigidly into sets, such as the set of lines extracted from a single image. In such cases, it is important for a visual module to be able to access subsets of these tokens, for example all the lines in the upper corner of an image or all the long lines in an image, examples of spatial and associative access.

Furthermore, it is often convenient for visual modules to operate on tokens as elements of a set. One form of this is when a user wants to combine multiple forms of access, for example by accessing the long lines in the upper corner of an image. Other times a visual module may need to operate over the tokens in a set, for example to find the average of the length of lines in a set. For these reasons, ISR includes operations for iterating over the tokens in a set, as well as taking the union, intersection and differences of sets³.

The Image Examiner is a very different kind of tool. It provides the visual modules designer with graphics support for displaying images and data. As such, it is only one of many tools for examining images, including displaying them at different scales or with altered color maps. What distinguishes the Image Examiner different from most other tools (e.g. XV) is that it is integrated with ISR and includes routines for displaying most common forms of visual symbolic tokens. It is therefore trivial, for example, to overlay a set of

² The data exchange format of the IUE reflects a similar belief.

³ The complement of a set is not well defined, since there is an infinite universe of possible tokens.

lines on top of an image, or on top of a symbolic image region. Tokens can also be pseudo-colored according to feature ranges, allowing the user to visually inspect symbolic data.

visual modules can be chained together simply by graphically connecting the output of one module to the input of another (see Figure 2). Loops and branching sequences of modules can also be graphically created. Adding a new module into the graphical interface is easily accomplished by tools

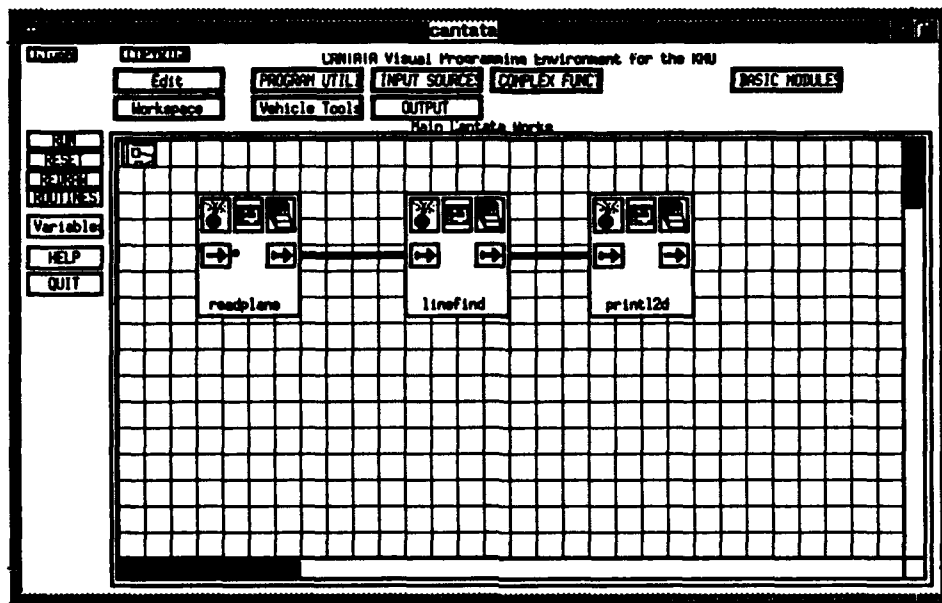


Figure 2. An Example Cantata Interface from Khoros

3.2. Khoros

Khoros [Argiro 91], although often compared to KBVision, is in fact a very different type of system, designed for different uses. Like KBVision, Khoros is organized around a set of visual modules. But whereas KBVision was designed as a high-level image understanding environment, with facilities for reasoning about lines, surfaces and other symbolic events, Khoros was designed as an image processing environment. As a result, it has no equivalent to the ISR and manipulates only image data. (It provides a set of tools which are similar in function to what the Image Examiner provides KBVision, but again they apply only to image data.)

Where Khoros is effective, however, is in its user interface. Khoros provides an execution environment, called *Cantata*, in which visual modules are represented by graphical icons. Clicking on an icon pops up a menu of the modules parameters, and allows the user to adjust those parameters without having to recompile the module or the system. Even more conveniently,

in the Khoros system called *ghostwriter*.

4. MoPLSE

Unfortunately, neither KBVision nor Khoros satisfied the needs of the MPL project. KBVision has a central data store (ISR) and convenient graphics tools, but does not provide a flexible enough graphical interface for sequencing modules or putting them in loops. Khoros has a flexible user interface, but only provides support for image-like data. Moreover, neither system was originally designed for real-time research, in that both systems pass data through files and execute each modules in its own UNIX process. If used in a real-time application, the costs of file access and process creation would prove unacceptable.

A new software environment was therefore needed for the MPL. MoPLSE was built around an improved version of the ISR database and Khoros's user interface, with graphics facilities built on the XV system developed at the University of Pennsylvania. What follows is a

brief description of MoPLSE in terms of these three components.

4.1. ISR3

The centerpiece of MoPLSE is the ISR. MoPLSE's ISR (called ISR3) is conceptually similar to the earlier versions built at the University of Massachusetts [Brolio 89] and to the ISR embedded in KBVision™. Both systems provide facilities for defining symbolic token types, adding new tokens to the database, and accessing tokens by name, value and spatial location (see above). ISR3 differs from the earlier versions, however, in that it stores native C data structures, is memory-based rather than file-based, and provides primitives for memory management and multiprocess synchronization. However, ISR3 does provide facilities for storing data to files for debugging.

Studying each of these improvements in turn, we begin with ISR3's ability to store native C structs rather than special, ISR-defined records. KBVision's ISR includes a language for defining token types. Beside the conceptual overhead of learning another syntax, this forces users to access fields of tokens through special ISR access functions, and at times to copy data into C structures. In a real-time environment, speed of data access is crucial, and copying data should be avoided. In ISR3, tokens are C structures stored in shared memory, and in order to define a new token type, a user simply adds a new structure definition to a file. By storing native C structures and returning pointers to them, ISR3 obviates the need to copy data (other than for making local destructive changes). Just as importantly, access to C structures is optimized by most commercial C compilers.

By default, ISR3 structures are stored not in a file but in shared memory. (Users can also specify that a token is local to a process and should be allocated in local memory.) As a result, when one process needs access to tokens created by another, it simply uses the ISR3 token access functions to get a pointer to the appropriate structures in memory. This is much faster than in KBVision, where one process has to write the data out to a file and the other has to read it back into its local memory. Moreover, ISR3 allows two processes to share data, which is not possible in KBVision.

Shared tokens in turn brings us to multiprocess synchronization. One problem with shared visual databases in a multiprocess environment is how to stop processes from accidentally overwriting or destroying each other's data. In a typical

application, hundreds or thousands of tokens may be in memory at any one time. If modules have uncontrolled access to these tokens and modify them, unpredictable interactions may cause hard-to-find bugs. On the other hand, it is not uncommon for a process to access a hundred tokens at a time, for example when a grouping routine looks for pencils of converging lines in vanishing point analysis. If such a process has to lock and unlock a token each time it reads a feature value, then given the relatively slow speed of semaphores under UNIX, protection becomes unacceptably expensive.

Our compromise, therefore, was to associate semaphores with sets of tokens. When a set of related tokens are created, for example the set of lines from a single image, a semaphore is allocated to protect those tokens. Any ISR3 function that accesses that set of tokens will first check the semaphore; if users access them surreptitiously through C pointers, they are expected to lock the semaphore before accessing the first token and to unlock it after the last token access. All ISR3 access function that create subsets of a set of tokens will assign the same semaphore to the subset that is used for the parent set.

Finally, ISR3 provides a level of memory management. For non-real-time, file-based systems, memory management is not a critical issue; for continuous, real-time systems, however, it is crucial. ISR3 applications operate in real-time loops, allocating new tokens on each iteration. Memory allocation must be rapid. More importantly, memory must be recycled, with space allocated to old tokens being reassigned to new ones once the old data is no longer needed. Unknown to the user, ISR3 provides buffers for tokens of every declared type, with freed tokens being returned to the appropriate buffer. For users who store tokens in hierarchies, ISR3 also provides functions for tracing through a hierarchy and freeing all the tokens in it, so that it is easy, for example, to free all the memory associated with a given image once that image is no longer current.

4.2. A Modified Cantata

MoPLSE's graphical user interface is a modification of the interface found in Khoros's Cantata program. As in Cantata, an icon is created for each visual module which, when clicked on, gives the user access to, and an ability to change, that module's parameters. A module can be executed simply by clicking on the icon's run button, and libraries of available modules can be selected with the mouse (see Figure 2).

More importantly, MoPLSE borrows Cantata's facilities for sequencing modules. If process B uses data created by process A, then the user simply draws a line from the output of A to the input of B. This tells the execution monitor to execute process A before process B, and to route its output accordingly. Users can also create infinite or counted loops in which B follows A and A follows B, or branching sequences of control in which the results of one module are used to select one of two control flow branches.

The problems with the original Cantata, as mentioned before, are 1) that it only provides facilities for passing image-like data from one module to the next, 2) it executes each module as a separate process, and 3) data is passed from one module to the next through files. MoPLSE addresses the first problem by integrating ISR3 into the execution controller. Modules may output a token or set of tokens of any type known to ISR3. Graphically, these tokens are passed to other modules just as image data is passed from module to module in Cantata - by connecting the output of one module to the input of another. The interface knows enough about ISR3 to check that the token types match, and if they do allows the connection to be made.

The ISR3 also solves Cantata's third problem of passing data through files. In MoPLSE, data is passed as pointers to tokens in shared memory, eliminating the time delay related to files⁴. The second drawback to Khoros is removed by having the execution environment execute visual modules as subroutines rather than separate processes. Although there are disadvantages associated with using subroutines, the gain in eliminating process creation overhead more than makes up for them.

4.3. Graphics

The third and final component of MoPLSE are its graphics tools. Khoros provides a complete set of graphics tools, but they are applicable only to image-like data. Facilities for drawing 2D line segments, for example, or projecting 3D lines are not provided. KBVision's Image Examiner includes the ability to display lines, regions and other symbolic tokens, but currently it cannot be expanded to display user-defined token types. It only displays the basic set of token types defined by Amerinex. Since many of the token types used

in research aboard MPL, such as surfaces and 3D lines, are not currently included among AAI's predefined token types, another set of graphics tools are included in MoPLSE.

Fortunately, many other research institutions have developed graphics tools before us. One such tool is XV, developed at the University of Pennsylvania. XV is a portable graphics facility, with available source code, for displaying images under X windows. Like Cantata's graphics tools, however, it is limited to displaying image-like data. In order to build a facility for displaying both standard and novel token types, we divided the graphics process into three steps, rasterization, overlay and display. MoPLSE has modules for rasterizing lines, regions, displacement vectors, and other types of tokens. These modules produce image-like data from symbolic tokens, and in some cases multiple modules exist for rasterizing a single type of token. Lines, for example, can be displayed with or without arrows indicating their gradient direction. Users who wish to display their own user-defined token types need to develop modules for rasterizing them according to their unique semantics. Another module is provided for overlaying rasterized data, so that, for example, lines can easily be displayed over the image they were extracted from. The overlay module is actually capable of any logical combination of two rasterized data images, but simple overlay is by far the most commonly used. Finally, a display module invokes XV to display the rasterized, and possibly overlaid, data.

5. References

- Argiro, J., KHOROS Documentation, University of New Mexico, 1991.
- Beveridge, J. R. and E. M. Riseman., "Can Too Much Perspective Spoil the View? A Case Study in 2D Affine and 3D Perspective Model Matching," Proc. of DARPA IUW, San Diego, CA., 1992.
- Brolio, J., Draper, B., Beveridge, J. R., and Hanson, A. R., ISR: A Database for Symbolic Processing in Computer Vision, IEEE Computer, Vol. 22, No. 12, December 1989, pp. 22-30.
- J.B. Burns, A.R. Hanson and E.M. Riseman, "Extracting Straight Lines," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.8, No.4, July 1986, pp.425-456.
- J. Dolan and E. Riseman, Computing Curvilinear Structure by Token-Based Grouping, IEEE CVPR, Champaign, IL, June 1992, pp.264-270

⁴ Caching techniques sometimes make files equivalent to passing pointers in memory, but this cannot be relied on, particularly with large data sets.

- Dolan, J. and R. Weiss. (1989). "Perceptual Grouping of Curved Lines," Proc. of DARPA IUW, Palo Alto, CA, pp. 1135-1145.
- Draper, B. (1989). "Integrating Top-Down Control with Intermediate-Level Vision," Proc. of SPIE Conference on Applications of AI - VII, Orlando, FL.
- Draper, B., J. Brolio, R. Collins, A. Hanson and E. Riseman. (1989). "The Schema System," IJCV, Vol. 2(3), pp. 209-250.
- R. Dutta, C. Weems, and E. Riseman, Parallel Dense Depth from Motion on the Image Understanding Architecture, Proc. 1993 DARPA IUW, Washington, DC, April 19-21, 1993.
- Fennema, Claude L., "Interweaving Reason, Action, and Perception," Ph.D. Thesis, Computer Science Department, University of Massachusetts, 1991; available as Technical Report 91-56.
- Kohler, R. R. and Hanson, A. R. "The VISIONS Image Operating System, Proc. 6th ICPR, Munich, Germany, October 1982.
- Kumar, R. Model Dependent Inference of 3D Information from a Sequence of 2D Images, Ph. D. Thesis and Technical Report TR92-04, Department of Computer Science, University of Massachusetts (Amherst), 1992.
- Pomerleau, D.A., Gowdy, J. and Thorpe, C.E. "Combining Artificial Neural Networks and Symbolic Processing for Autonomous Robot Guidance," Proc. of the DARPA Image Understanding Workshop, San Diego, CA, Jan. 1992. pp. 961--967.
- Pomerleau, D. A., Neural Network Based Autonomous Navigation, in Charles E. Thorpe, Ed., *Vision and Navigation: The Carnegie Mellon Navlab*, Chapter 5. Kluwer Academic Publishers, 1990.
- H. Sawhney, R. Kumar, A. Hanson, E. Riseman, Landmark-Based Navigation - Model Extension and Refinement, Proc. DARPA IUW, Washington, DC, April 19-21, 1993.
- Sawhney, H. and A. R. Hanson, Tracking, Detection, and 3D Representation of Potential Obstacles using Affine Constraints, Proc. of DARPA IUW, San Diego, CA, 1992a, pp. 1009-1017.
- Sawhney, Harpreet, Spatial and Temporal Grouping in the Interpretation of Image Motion, Ph.D. Thesis, Computer Science Department, University of Massachusetts, 1992; also Technical Report 92-05.
- J. Thomas, A. Hanson, and J. Oliensis, Understanding Noise : The Critical Role of Motion Error in Scene Reconstruction, Proc. ICCV, Berlin, Germany, May 11-13, 1993a, to appear; (A similar paper appears in these proceedings).
- J. Thomas, "Obtaining the Robot Path Using Automatically Acquired Models, Proc. International Conference on Intelligent Autonomous Systems (IAS), Pittsburgh, PA, February 1993b, to appear.
- Weems, C.C., Levitan, S.P., Hanson, A.R., Riseman, E.M., Shu, D.B., Nash, J.G., The Image Understanding Architecture, Intl. Journal of Computer Vision, 2, 251-282 (1989), Kluwer Academic Publishers, Boston, MA.
- Williams, L. R. and A. R. Hanson. (1988). "Translating Optical Flow into Token Matches and Depth from Looming," Proc. of IEEE ICCV, Tarpon Springs, FL, pp. 441-448.
- Williams, T. Image Understanding Tools, Proc. 10th ICPR, Atlantic City, NJ, June 1990, pp. 606-610.